CURSO PRÁTICO 34 DE PROGRAMAÇÃO DE COMPUTADORES

INPUT

PROGRAMAÇÃO BASIC - PROGRAMAÇÃO DE JOGOS - CÓDIGO DE MÁQUINA

NOVA CULTURAL

Cz$ 65,00

INPUT

Vol. 3 Nº 34

NESTE NÚMERO

PROGRAMAÇÃO BASIC

GRÁFICOS DA ROM NO SPECTRUM (2)

Utilização de caracteres gráficos dentro do programa. Linhas longas. A função **TO** 661

PROGRAMAÇÃO DE JOGOS

UM JOGO DE ESTRATÉGIA

Objetivos do jogo. A primeira tela. Saldos e custos. Opções do jogador..... 662

PROGRAMAÇÃO BASIC

SPRITES PARA O TRS-80 (3)

Os caracteres de controle do cursor. Figuras complexas. Animação de figuras complexas ... 669

PROGRAMAÇÃO BASIC

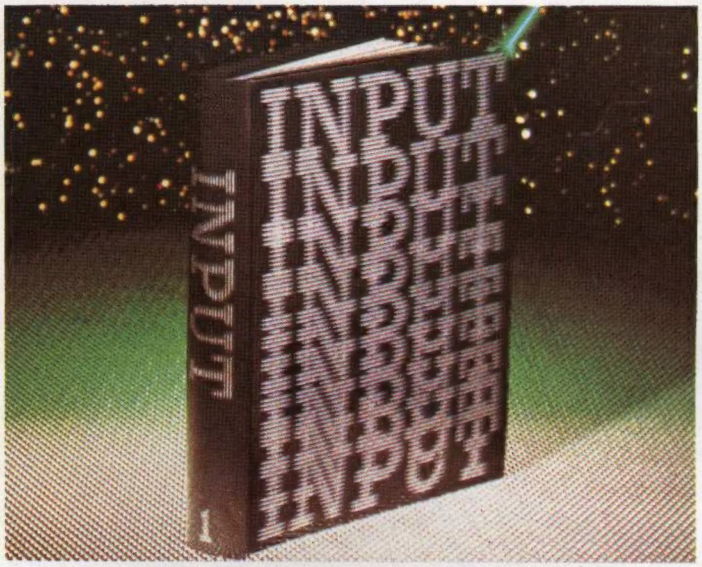
SIMULAÇÃO: FAÇA A BOLA ROLAR

O modelo da bola. Uma mira móvel. Como a equação funciona. Aperfeiçoe os programas. 670

CÓDIGO DE MÁQUINA

UM ASSEMBLER PARA O TRS-80

O programa do MSX modificado. Funcionamento do Assembler. Como utilizar o programa. 679



PLANO DA OBRA
INPUT é uma obra editada em fascículos semanais, e cada conjunto de 15 fascículos compõe um volume. A capa para encadernação de cada volume estará à venda oportunamente.

FÉRIAS, VIAGENS, MUDANÇAS...
NÃO FIQUE COM A COLEÇÃO INCOMPLETA
Se você está saindo de férias, pretende viajar ou vai se ausentar por algum tempo, avise antecipadamente seu jornaleiro. Ele pode guardar os seus fascículos enquanto você estiver fora. Se, por qualquer motivo, você perdeu alguns números, peça-os também a seu jornaleiro, ou entre em contato com nossa Distribuidora:

1. **Pessoalmente** — Em *São Paulo*, os endereços são: Rua Brigadeiro Tobias, 773, Centro, telefone 227-4188; Av. Industrial, 117, Santo André, telefone 449-0411, das 7h30 às 17h00 - dias úteis. No *Rio de Janeiro*, Av. Mem de Sá, 191/193, Centro, telefone (021) 222-7422, das 7h30 às 17h00 - dias úteis.
2. **Por carta** — Envie para:
   DINAP — Distribuidora Nacional de Publicações
   Números Atrasados
   Estrada Velha de Osasco, 132 — Jardim Teresa
   CEP 06040 — Osasco — SP
3. **Por telex** — Utilize o nº (011) 33 670 DNAP.

Em **Portugal**, os pedidos devem ser feitos à Distribuidora Jardim de Publicações Lda. — Qta. Pau Varais, Azinhaga de Fetais, 2685, Camarate, Lisboa; Apartado 57; Telex 43 069 JARLIS P.
Atenção: Após seis meses do encerramento da coleção, o atendimento dos pedidos dependerá da disponibilidade do estoque.
Obs.: Quando pedir livros, mencione sempre o título e/ou autor da obra, além do número da edição.

COLABORE CONOSCO
Encaminhe seus comentários, críticas, sugestões ou reclamações ao
SERVIÇO DE ATENDIMENTO AO LEITOR
Caixa Postal 9 442, São Paulo — SP.

EDITOR
RICHARD CIVITA
NOVA CULTURAL

Presidente
Flávio Barros Pinto
Diretoria
Carmo Chagas, Iara Rodrigues,
Pierluigi Bracco, Plácido Nicoletto,
Roberto Silveira, Shozi Ikeda,
Sônia Carvalho

REDAÇÃO
Diretor Editorial: Carmo Chagas
Editores Executivos:
Berta Sztark Amar, Stefania Crema
Editor Chefe: Paulo de Almeida
Editoras Assistentes: Ana Lúcia B. de Lucena,
Marisa Soares de Andrade
Chefe de Arte: Carlos Luiz Batista
Assistentes de Arte: Dagmar Bastos Sampaio,
Grace Alonso Arruda, Monica Lenardon Corradi

Secretário de Redação: Mauro de Queiroz

Colaboradores
Consultor Editorial Responsável:
Dr. Renato M. E. Sabbatini
(Diretor do Núcleo de Informática Biomédica da Universidade Estadual de Campinas-SP)
Execução Editorial: DATAQUEST Assessoria em Informática Ltda., Campinas
Tradução, adaptação, programação e redação:
Abílio Pedro Neto, Aluísio J. Dornellas de Barros,
Marcelo R. Pires Therezo, Marcos Huascar Velasco,
Raul Neder Porrelli, Ricardo J. P. de Aquino Pereira
Coordenação Geral: Rejane Felizatti Sabbatini

COMERCIAL
Diretor Comercial: Roberto Silveira
Gerente Comercial: Joaquim Celestino da Silva
Gerente de Circulação: Denise Mozol
Gerente de Propaganda e Publicidade: José Carlos Madio
Gerente de Pesquisa e Análise de Mercado:
Wagner M. P. Nabuco de Araújo

CLC

A Editora Nova Cultural Ltda. é uma empresa do Grupo CLC — Comunicações, Lazer e Cultura S.A.
Presidente: Richard Civita
Diretoria: Flávio Barros Pinto, João Gomez,
Menahem M. Politi, Renê C.X. Santos,
Stélio Alves Campos

© Marshall Cavendish Limited, 1984/85.
© Editora Nova Cultural Ltda., São Paulo, Brasil, 1986; 2ª edição, 1987.
Edição organizada pela Editora Nova Cultural Ltda.
Av. Brigadeiro Faria Lima, 2000 - 3º andar
CEP 01452 - São Paulo - SP - Brasil
(Artigo 15 da Lei 5 988, de 14/12/1973).
Esta obra foi composta pela AM Produções Gráficas Ltda.
e impressa pela Companhia Lithographica Ypiranga.

GRÁFICOS DA ROM NO SPECTRUM (2)

- CARACTERES GRÁFICOS DENTRO DO PROGRAMA
- A FUNÇÃO **CHR$**
- LINHAS LONGAS
- A FUNÇÃO **TO**

Em artigo anterior, explicamos o que são os caracteres gráficos e como entrá-los diretamente pelo teclado. Trataremos aqui de sua utilização dentro de programas.

Para especificar caracteres gráficos dentro de um programa, sem precisar digitá-los diretamente (veja artigo publicado à página 640), emprega-se a utilíssima função **CHR$**. Para imprimir na tela um quadrado vermelho, podemos usar, por exemplo:

```
PRINT INK 2,CHR$(143)
```

O programa seguinte mostra a tabela de correspondência entre códigos numéricos e gráficos na tela dos micros da linha Spectrum:

```
10 CLS
20 FOR j=128 TO 143 STEP 5
30 FOR i=j TO j+4
40 PRINT i;" ";CHR$(i);" ";
50 NEXT i
60 PRINT
70 NEXT j
```

Os caracteres são impressos ordenadamente em fileiras, na tela, por meio dos dois laços que começam nas linhas 20 e 30 do programa. O primeiro laço varia **J** de 193 a 242 (a faixa de códigos correspondente aos caracteres gráficos), de 6 em 6. O laço seguinte percorre todos os valores numéricos existentes entre **J** e **J+5**.

O **PRINT** da linha 60 serve para encerrar uma fileira de seis códigos e suas representações gráficas. Estas são exibidas pela linha 40.

LINHAS LONGAS

Se você pretende empregar caracteres gráficos com certa freqüência em um programa, convém armazenar seus códigos em uma variável alfanumérica. Assim, ficará bem mais fácil utilizá-los, e você não terá necessidade de consultar a todo momento a tabela de códigos gráficos, quando estiver desenvolvendo o programa.

Da mesma maneira, quando se pretende utilizar uma cadeia de caracteres gráficos em vários pontos de um programa, é interessante armazená-los em uma variável alfanumérica. Como o Spectrum não tem a função **STRING$**, que permite fazer isto com um único comando, será preciso escrever um pequeno laço de acumulação:

```
10 CLS
20 PRINT AT 19,0;"Codigo
Grafico (128-143) : "
25 INPUT c
30 PRINT AT 19,0;"Comprimento
(1-29) : "
40 INPUT n
50 LET s$=""
60 FOR i=1 TO n
70 LET s$=s$+CHR$(c)
80 NEXT i
90 PRINT AT 10,1;s$
95 PAUSE 100
100 GOTO 10
```

A cadeia alfanumérica **S$** tem **L** códigos, que são exibidos na tela ao mesmo tempo, quando se emprega o comando **PRINT S$**. Este truque será muito útil quando você precisar usar com freqüência, no programa, linhas de separação, molduras etc.

A FUNÇÃO TO

A função **TO** tem grande utilidade na construção de gráficos com símbolos de teclado, pois ela permite que se extraiam segmentos curtos de uma cadeia gráfica maior.

Suponhamos que você precise usar retas horizontais de diferentes tamanhos, na composição de uma tela gráfica. Em vez de gerar várias cadeias gráficas, usando a técnica do laço **FOR...NEXT** explicada anteriormente, você poderá gerar uma única cadeia, com comprimento máximo (uma cadeia **C$**, com 32 caracteres, por exemplo).

Depois, para colocar em um ponto qualquer da tela do micro uma cadeia gráfica de doze caracteres, bastará utilizar o **PRINT AT** combinado com **C$(TO 12)**. A função **TO**, no caso, serve para extrair os primeiros doze caracteres da cadeia **C$**, simplificando muito o processo.

MICRO DICAS

CARACTERES INVERTIDOS

Em alguns tipos de computador, o conjunto de caracteres gráficos — isto é, aqueles caracteres que podem ser obtidos pressionando-se <**SHIFT**><**9**> — abrange também os chamados caracteres invertidos.

Estes nada mais são do que uma representação "em negativo" da forma usual de exibição na tela. As duas formas — "normal" e "invertida" — aparecerão diferentemente conforme a marca do computador.

Nos micros originais da Sinclair (o ZX-81 e seu equivalente norte-americano, o Timex), por exemplo, a representação normal de vídeo compõe-se de caracteres pretos sobre fundo claro. O caractere invertido seria, portanto, branco sobre fundo preto. Como o fundo normal de tela é branco, os caracteres invertidos aparecem como um retângulo preto, com o caractere impresso em branco. Já a cor clara do fundo dos caracteres em vídeo normal fica igual ao fundo da tela. Esse tipo de representação de vídeo é característico dos modelos nacionais da Microdigital (TK-82C, TK-83 e TK-85).

Alguns outros computadores nacionais, entre eles o CP-200, da Prológica, têm tela com fundo preto, normalmente, e exibem caracteres claros. Aqui, o caractere invertido, obtido com <**SHIFT**><**9**>, é escuro sobre fundo claro.

Existem, ainda, alguns modelos que permitem ao usuário mudar de vídeo preto para branco, conforme sua conveniência, por meio de um interruptor colocado no console.

Os caracteres invertidos têm diversas aplicações. São utilizados, particularmente, para realçar determinados títulos, rótulos ou mensagens numa tela. Alguns deles — como ponto, dois pontos, asterisco, barra e a letra O — podem ser incorporados como elementos gráficos na composição de um desenho.

UM JOGO DE ESTRATÉGIA

Entre no mundo dos grandes riscos — e grandes lucros. Você é dono de uma empresa de mineração. Terá astúcia e talento para tomar decisões e coragem para levá-las a cabo?

Neste jogo de estratégia empresarial, o jogador desempenha o papel de dono de uma companhia de mineração. Seu dever é cuidar para que o empreendimento prospere ao máximo. Como durante o jogo muitas opções lhe são oferecidas, a sorte da empresa dependerá de sua capacidade de tomar decisões.

Jogos de estratégia, assim como de aventuras, são geralmente escritos em BASIC, não havendo necessidade de se recorrer a rotinas em código. Como eles também não incluem longos textos e comentários, são compatíveis com micros de pouca quantidade de memória. Embora nosso jogo tenha sido enriquecido com rotinas gráficas que ilustram o processo de mineração — o que aumentou o espaço de memória utilizado —, o programa cabe nos micros de 16K.

Apresentaremos o programa em duas partes, pois ele é bastante longo. Trataremos aqui de sua parte central, mas algumas das rotinas essenciais para fazê-lo funcionar só serão abordadas no próximo artigo.

Depois de digitar esta primeira seção, grave o programa e aguarde a publicação da parte que o completa. Em alguns computadores, se o programa for executado neste estágio, a tela será preenchida com diversas informações sobre o *status* do jogo, além de uma série de opções. Contudo, surgirá também uma mensagem de erro, uma vez que o programa está incompleto.

OS OBJETIVOS DO JOGO

Quando o jogo começa, o jogador tem um saldo a seu favor — a companhia de mineração e 2 milhões em dinheiro. Sua tarefa é investir estes recursos inteligentemente, na busca do precioso metal. O objetivo do jogo é reunir a maior quantia de dinheiro possível em trinta rodadas. O jogo permite que uma pessoa jogue sozinha ou que duas joguem ao mesmo tempo.

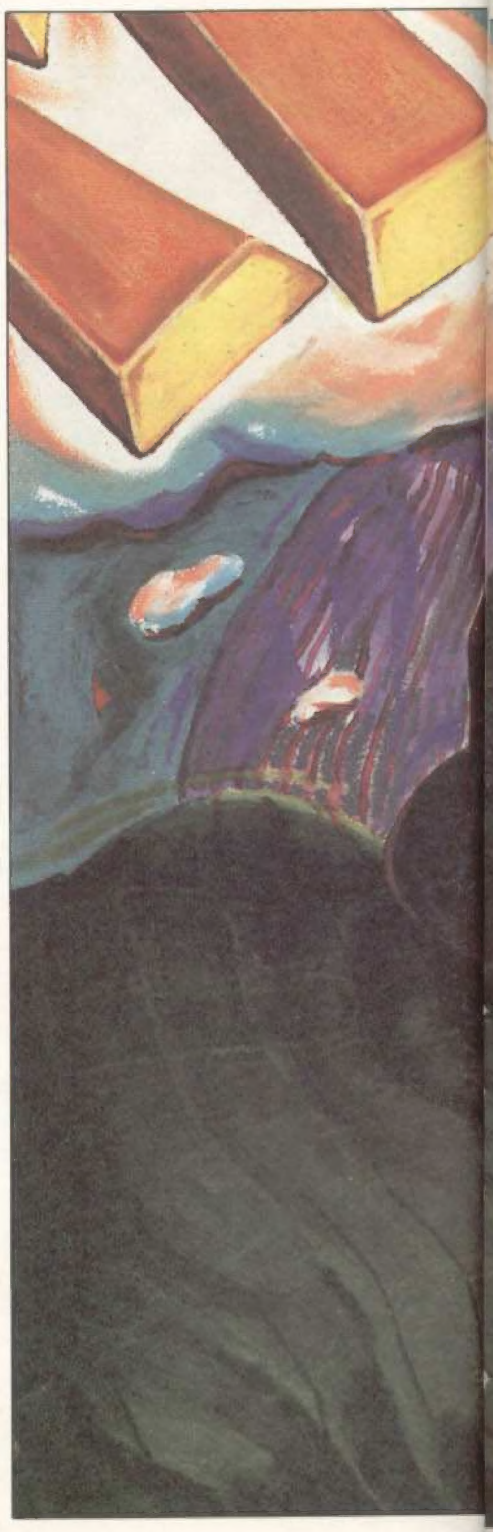
A cada lance, o jogador se vê diante de várias opções. Antes de começar as escavações, precisa encontrar um local promissor; para isso, deverá destinar certa quantia à contratação de um geólogo que lhe forneça uma avaliação profissional. O jogador será, então, informado sobre suas chances de encontrar ouro e da provável profundidade e dimensões do veio. A responsabilidade de decidir se vale a pena explorar determinada mina é sua.

Como o trabalho de escavação e garimpagem envolve altas somas, poderá ser conveniente investir na pesquisa e desenvolvimento de equipamentos que reduzam os custos da operação. Ou talvez seja melhor começar logo as escavações — só o jogador pode decidir. Se ele iniciá-las, gráficos coloridos ilustrarão o progresso do trabalho. Caso não encontre ouro, precisará resolver se vale a pena continuar a prospecção ou se é melhor abandonar a mina e partir para outra.

No decorrer do jogo, dois outros fatores influem nos resultados. Uma vez encontrado o ouro, pode-se guardá-lo em cofre-forte ou vendê-lo no mercado. É razoável guardar o metal, se não houver necessidade imediata de dinheiro; pode ser vantajoso esperar por uma boa cotação para vendê-lo — a cotação flutua durante o jogo. Contudo, armazenar ouro é arriscado, pois há ladrões por toda parte e, quanto maior a quantidade, maior a tentação.

A segunda parte do programa cuidará dos detalhes do funcionamento do jogo. Agora, digite a primeira parte.

```
 5 BORDER 6: PAPER 6: INK 0:
 CLS
10 PRINT AT 9,2;"Quantos joga
dores? (1 ou 2)": LET a$=
INKEY$: IF a$="" THEN  GOTO
10
20 IF a$<"1" OR a$>"2" THEN
GOTO 10
30 LET p=VAL a$: LET nop=p
40 DIM a(2,6): DIM c(2,5):
DIM a$(p,8): DIM r(2): LET er
=10000
50 LET r(1)=0: LET r(2)=0:
LET a(1,1)=2000000: LET a(1,2
)=2000000: LET a(2,1)=2000000
: LET a(2,2)=2000000: LET a(1
,3)=0: LET a(2,3)=0: LET a(1,
4)=100000: LET a(2,4)=100000:
 LET a(1,5)=0: LET a(2,5)=0:
LET a(1,6)=0: LET a(2,6)=0:
```



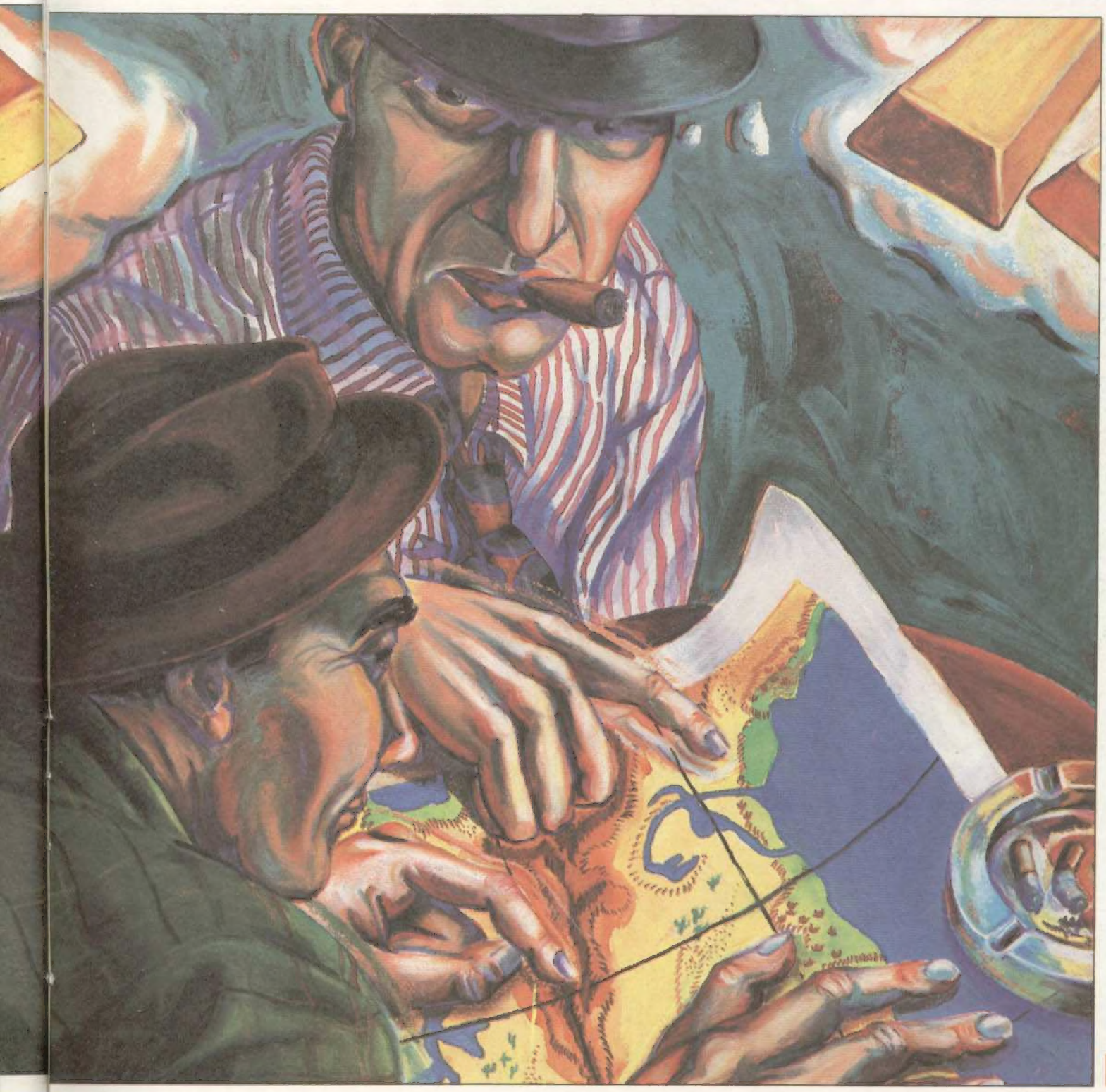
- OBJETIVOS DO JOGO
- UM OU DOIS JOGADORES
- A PRIMEIRA TELA
- SALDOS E CUSTOS
- COMO DAR INFORMAÇÕES

AO PROGRAMA

- OFERTA DE OPÇÕES

AO JOGADOR

- ROTINA DOS LADRÕES DE OURO
- EFEITOS SONOROS



```
 PRINT
 70 FOR n=1 TO p: INPUT "Nome
 do jogador ";(n);"?", LINE a$
 (n): NEXT n
200 FOR n=1 TO 30: FOR m=1 TO
nop
202 BORDER 7: PAPER 7: INK 0:
CLS
210 PRINT  PAPER 6;n: PRINT
PAPER 1; INK 6;AT 0,4;" M I N
A   D E  O U R O "
220 PRINT 'TAB 16;a$(1);: IF n
op=2 THEN  PRINT TAB 24;a$(2);
230 PRINT '"SALDO TOTAL  $";
TAB 15;a(1,1);: IF nop=2 THEN
PRINT TAB 24;a(2,1);
240 PRINT '"DINHEIRO     $";
TAB 15;a(1,2);: IF nop=2 THEN
PRINT TAB 24;a(2,2);
250 PRINT '"OURO kg";TAB 15;a(
1,3);: IF nop=2 THEN  PRINT
TAB 24;a(2,3);
260 PRINT '"CUSTO P/CAVAR$";
TAB 15;a(1,4);: IF nop=2 THEN
PRINT TAB 24;a(2,4);
270 PRINT '"No. DE MINAS";TAB
15;a(1,5);: IF nop=2 THEN
PRINT TAB 24;a(2,5);
280 PRINT '"PROFUNDID. m";TAB
15;a(1,6);: IF nop=2 THEN
PRINT TAB 24;a(2,6);
300 PRINT '' PAPER 4; INK 0;"C
otacao do ouro no mercado:":
PRINT "$";er;" por kg de ouro"
400 PRINT ' PAPER 5;">-";a$(m)
500 PRINT  PAPER 2; INK 7;"1";
: PRINT "-Pesquisa e desenvolv
imento"
510 PRINT  PAPER 2; INK 7;"2";
: PRINT "-Levantamento geologi
co"
520 PRINT  PAPER 2; INK 7;"3";
: PRINT "-Cavar mais 200 m"
530 PRINT  PAPER 2; INK 7;"4";
: PRINT "-Vender ouro no merca
do"
540 PRINT  PAPER 2; INK 7;"5";
: PRINT "-Passa a vez"
550 PRINT : PRINT  FLASH 1;
PAPER 1; INK 6;"Faca sua opcao
 "
600 LET i$=INKEY$: IF i$=""
THEN  GOTO 600
610 IF i$<"1" OR i$>"5" THEN
GOTO 600
620 GOSUB VAL i$*1000
700 IF a(m,2)<0 THEN  GOTO
 7000
710 LET er=er+INT (RND*1000)-
 200
720 IF INT (RND*1600)-a(m,3)<0
THEN  GOSUB 900
740 LET a(m,1)=a(m,2)+a(m,3)*
er
750 PAPER 7: INK 0: BORDER 7:
CLS
790 NEXT m
800 NEXT n
810 PAPER 5: BORDER 5: INK 0:
CLS
820 PRINT  FLASH 1; INK 7;
PAPER 2;AT 6,10;" FIM DO JOGO"
```

```
830 PRINT 'p5;"Saldo total de
";a$(1): PRINT TAB 11;"$";a(1,
1)
840 IF nop=2 THEN  PRINT 'TAB
5;"Saldo total de ";a$(2):
PRINT TAB 11;"$";a(2,1)
850 PRINT '' PAPER 2; INK 6;
FLASH 1;TAB 1;"Qualquer tecla
para recomecar"
860 IF INKEY$<>"" THEN  GOTO
 860
870 IF INKEY$="" THEN  GOTO
 870
880 RUN
900 PAPER 2: INK 6: BORDER 2:
 CLS
905 LET jk=INT (RND*100)+50:
IF jk>a(m,3) THEN  LET jk=a(m,
3)
910 PRINT  PAPER 6; INK 1;
FLASH 1;AT 9,10;" R O U B O "
920 PRINT : PRINT  INK 7;"
Foram roubados ";jk;"kg de":
PRINT "    seu ouro": LET a(m,
3)=a(m,3)-jk: LET a(m,1)=a(m,1
)-(jk*er)
930 FOR x=1 TO 35: SOUND .05,
40: SOUND .05,20: NEXT x
940 BORDER 7: PAPER 7: INK 0:
CLS : RETURN
```

```
20 DIM A(1,5),C(1,4)
90 SCREEN 1:COLOR 1,10,4:KEY OF
F
100 VPOKE BASE(6),4*16+4:VPOKE
BASE(6)+1,6*16+6
110 FOR I=6 TO 7:VPOKE BASE(6)+
I,15*16+6:NEXT
120 LOCATE 0,5:PRINT "Quantos j
ogadores (1 ou 2) ?;
130 A$=INKEY$:IF A$<"1" OR A$>"
2" THEN 130
140 P=VAL(A$):NO=P:ER=10000:A(0
,0)=2000000:A(0,1)=A(0,0):A(1,
0)=A(0,0):A(1,1)=A(0,0):A(0,3)=
100000!:A(1,3)=A(0,3)
150 FOR N=1 TO P:PRINT:PRINT:PR
INT"Nome do jogador";N;:LINE IN
PUT A$(N-1):IF LEN(A$(N-1))>8 T
HEN A$(N-1)=LEFT$(A$(N-1),8)
160 NEXT
200 FOR N=0 TO 29:FOR M=0 TO NO
-1
202 FOR I=0 TO 767:VPOKE BASE(5
)+I,0:NEXT:LOCATE 0,0
204 FOR I=0 TO 31:VPOKE BASE(5)
+I,8:VPOKE BASE(5)+736+I,8:NEXT
206 FOR I=0 TO 24:VPOKE BASE(5)
+I*32,8:VPOKE BASE(5)+31+I*32,8
:NEXT
210 LOCATE 9,0:PRINT "MINA DE O
URO"
220 LOCATE 12,2:PRINT A$(0);:IF
 NO=2 THEN LOCATE 21,2:PRINT A$
(1)
230 LOCATE 0,3:PRINT "SALDO TOT
AL";TAB(11);A(0,0);:IF NO=2 THE
N LOCATE 20,3:PRINT A(1,0)
240 LOCATE 0,4:PRINT "DINHEIRO"
;TAB(11);A(0,1);:IF NO=2 THEN L
OCATE 20,4:PRINT A(1,1)
250 LOCATE 0,5:PRINT "OURO";TAB
(11);A(0,2);:IF NO=2 THEN LOCAT
E 20,5:PRINT A(1,2)
260 LOCATE 0,6:PRINT "$ P/ CAVA
R";TAB(11);A(0,3);:IF NO=2 THEN
 LOCATE 20,6:PRINT A(1,3)
270 LOCATE 0,7:PRINT "NO DE MIN
AS";TAB(11);A(0,4);:IF NO=2 THE
N LOCATE 20,7:PRINT A(1,4)
280 LOCATE 0,8:PRINT "PROFUNDI/
/";TAB(11);A(0,5);:IF NO=2 THEN
 LOCATE 20,8:PRINT A(1,5)
300 LOCATE 1,10:PRINT "COTAÇÃO
DO OURO NO MERCADO":LOCATE 3,11
:PRINT ER;"POR KG DE OURO"
400 LOCATE 12,13:PRINT A$(M)
500 PRINT:PRINT "1-Pesquisa e d
esenvolvimento"
510 PRINT "2-Levantamento geoló
gico    "
520 PRINT "3-Cavar mais 200 m
    "
530 PRINT "4-Vender ouro no mer
cado    "
540 PRINT "5-Passa a vez
    "
600 A$=INKEY$:IF A$<"1" OR A$>"
5" THEN 600
620 ON VAL(A$) GOSUB 1000,2000,
3000,4000,5000
700 IF A(M,1)<0 THEN 7000
710 ER=ER+1000*INT(RND(1))-200
720 IF RND(1600)-A(M,2)<0 GOSUB
 900
740 A(M,0)=A(M,1)+A(M,2)*ER
750 CLS
790 NEXT M,N
810 CLS
820 LOCATE (5,5):PRINT "FIM DE
JOGO"
830 LOCATE (0,7):PRINT "SALDO T
OTAL DE";A$(0);TAB(23);A(0,0)
840 IF NO=2 THEN PRINT "SALDO T
OTAL DE";A$(1);TAB(23);A(1,0)
850 PRINT:PRINT "QUALQUER TECLA
 PARA RECOMEÇAR"
860 IF INKEY$="" THEN 860 ELSE
RUN
900 CLS
905 JK=INT(RND(1)*100)+50:IF JK
>A(M,2) THEN JK=A(M,2)
910 PRINT TAB(9);"R O U B O"
920 PRINT:PRINT "    FORAM ROUB
ADOS";JK;"KG":PRINT "DE SEU OUR
O":A(M,2)=A(M,2)-JK:A(M,0)=A(M,
0)-JK*ER
930 PLAY"T255O1CDEFBAGFED"
940 FOR I=1 TO 500:NEXT:CLS:RET
URN
```

```
20  DIM A(1,5),C(1,4),A$(1)
120  HOME : PRINT "QUANTOS JOGA
DORES (1 OU 2) ?";
130  GET A$: IF A$ < "1" OR A$
> "2" THEN 130
140 P =  VAL (A$):NO = P:ER = 1
0000:A(0,0) = 2000000:A(0,1) =
A(0,0):A(1,0) = A(0,0):A(1,1) =
A(0,0):A(0,3) = 10000:A(1,3) =
A(0,3)
150  FOR N = 1 TO P: PRINT : PR
INT : PRINT "NOME DO JOGADOR ";
N;: INPUT A$(N - 1): IF  LEN (A
$(N - 1)) > 8 THEN A$(N - 1) =
 LEFT$ (A$(N - 1),8)
160  NEXT
200  FOR N = 0 TO 29: FOR M = 0
 TO NO - 1
202  HOME
210  INVERSE : HTAB 12: PRINT "
 MINA DE OURO ": NORMAL
220  PRINT : PRINT  TAB( 20);A$
(0);: IF NO = 2 THEN  PRINT  TA
B( 30);A$(1);
230  PRINT : PRINT "SALDO TOTAL
"; TAB( 20);A(0,0);: IF NO = 2
THEN  PRINT  TAB( 30);A(1,0);
240  PRINT : PRINT "DINHEIRO";
TAB( 20);A(0,1);: IF NO = 2 THE
N  PRINT  TAB( 30);A(1,1);
250  PRINT : PRINT "OURO   KG";
 TAB( 20);A(0,2);: IF NO = 2 TH
EN  PRINT  TAB( 30);A(1,2);
260  PRINT : PRINT "CUSTO P/ CA
VAR  $"; TAB( 20);A(0,3);: IF N
O = 2 THEN  PRINT  TAB( 30);A(1
,3);
270  PRINT : PRINT "N.O DE MINA
S"; TAB( 20);A(0,4);: IF NO = 2
 THEN  PRINT  TAB( 30);A(1,4);
```

Como assegurar bons resultados na procura de ouro?

Quando o programa estiver completo, você verá que ele segue as regras do comércio — atividade que, entre outras coisas, inclui sorte. Por isso mesmo, é difícil dar a receita do sucesso e da fortuna. Mas algumas dicas terão utilidade.

Embora os custos de mineração possam ser reduzidos graças a pesquisa e desenvolvimento, não é aconselhável investir grandes somas neste item, já que você conta só com trinta rodadas para enriquecer.

Faça escavações somente em minas com boa possibilidade de produzir ouro, a uma pequena profundidade. Não fique, porém, eternamente passando a vez, pois o jogo pode acabar antes que a mina ideal apareça.

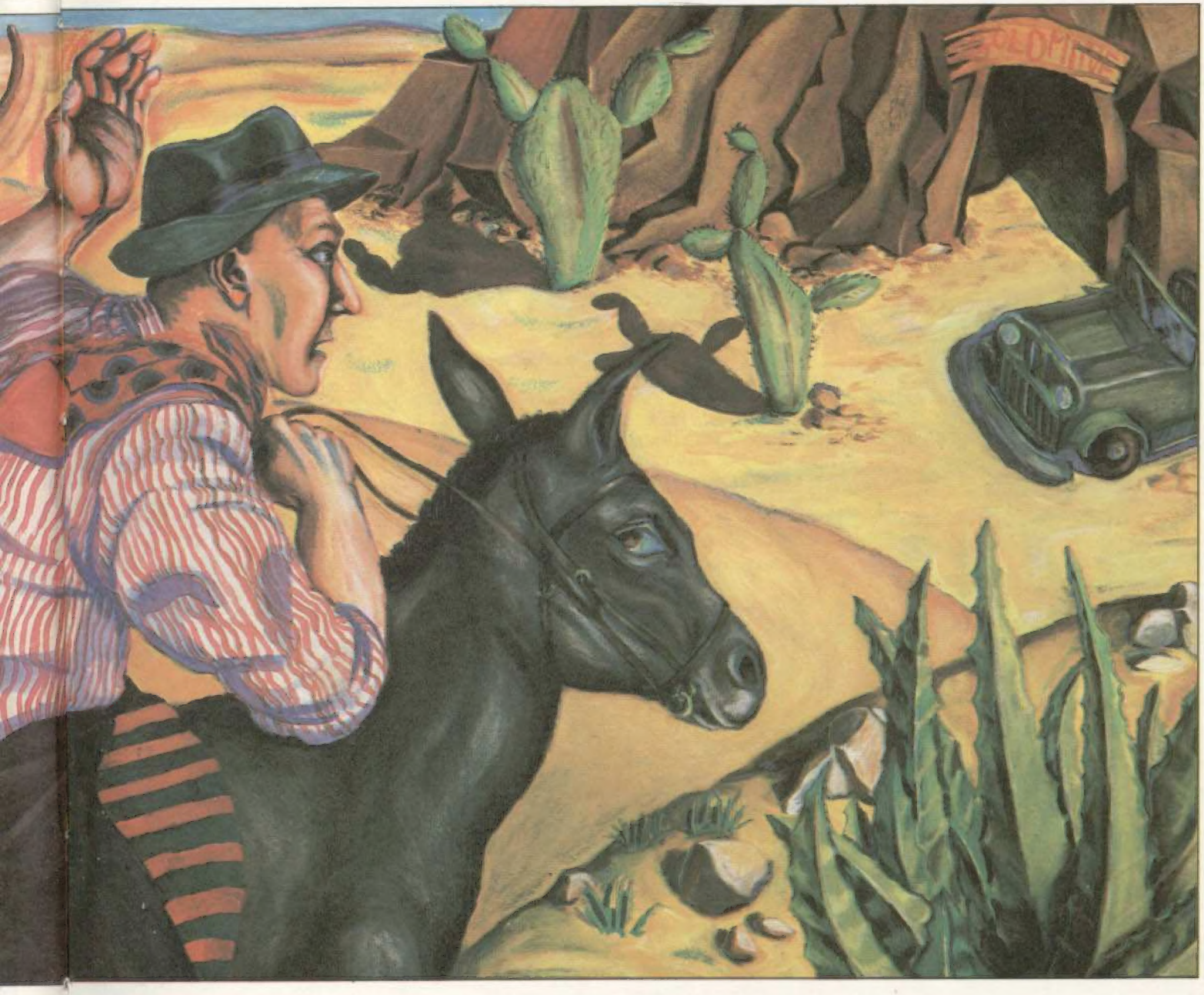
Se estocar ouro, você correrá o risco de ser roubado, mas compare as chances de que isso ocorra com a cotação do ouro vigente no mercado — tente sempre vender o metal em períodos de alta.

```
280  PRINT : PRINT "PROFUNDIDAD
E   M"; TAB( 20);A(0,5);: IF NO
 = 2 THEN  PRINT  TAB( 30);A(1,
5);
300  PRINT : PRINT : PRINT "COT
ACAO DO OURO :": PRINT ER;" POR
 KG DE OURO"
400  PRINT : PRINT  TAB( 14);A$
(M)
500  PRINT : PRINT "1- PESQUISA
 E DESENVOLVIMENTO"
510  PRINT "2- LEVANTAMENTO GEO
LOGICO"
520  PRINT "3- CAVAR MAIS 200 M
"
530  PRINT "4- VENDER OURO NO M
ERCADO"
540  PRINT "5- PASSAR A VEZ"
600  GET A$: IF A$ < "1" OR A$
> "5" THEN 600
620  ON  VAL (A$) GOSUB 1000,20
00,3000,4000,5000
700  IF A(M,1) < 0 THEN 7000
710 ER = ER +  INT ( RND (1) *
1000) - 500
720  IF  INT ( RND (1) * 1600)
- A(M,2) < 0 THEN  GOSUB 900
740 A(M,0) = A(M,1) + A(M,2) *
ER
750  HOME
790  NEXT M,N
810  HOME
820  INVERSE : HTAB 14: PRINT "
 FIM DE JOGO ": NORMAL
830  VTAB 5: PRINT "SALDO TOTAL
 DE ";A$(0);" : ";A(0,0)
840  IF NO = 2 THEN  PRINT : PR
INT "SALDO TOTAL DE ";A$(1);" :
 ";A(1,0)
850  PRINT : PRINT : PRINT "QUA
LQUER TECLA PARA RECOMECAR"
860  GET A$: RUN
900  HOME
905 JK =  INT ( RND (1) * 100)
+ 49: IF JK > A(M,2) THEN JK =
A(M,2)
910  INVERSE : HTAB 15: PRINT "
 R O U B O ": NORMAL
920  PRINT : PRINT "FORAM ROUBA
DOS ";JK;" KG DE SEU OURO":A(M,
2) = A(M,2) - JK:A(M,0) = A(M,0
) - JK * ER
930  FOR I = 1 TO 2000: NEXT
940  HOME : RETURN
```

T

```
10 PMODE 3,1:CLS
20 DIM H(23),T(0),D(1),B(1),A(1
,5),C(1,4),A$(1),R(1)
40 FOR K=0 TO 9:READ NU$(K):NEX
T
50 DATA NR2D4R2U4BR2,BFEND4BR2,
R2D2L2D2R2BU4BR2,NR2BD2NR2BD2R2
U4BR2,D2R2D2U4BR2,NR2D2R2D2L2BE
4,D4R2U2L2BE2BR2,R2ND4BR2,NR2D4
R2U2NL2U2BR2,NR2D2R2D2U4BR2
60 PCLS 3
70 DRAW"BM36,23C2L35U6E3R3NU4R5
U10E2RE3R3F4D3F4DF2DF2DF3D2":PA
INT(18,16),2
80 DRAW"BM24,9C3G2D6F5R3E2UH2UH
UH2UH2BM20,1NLDL2GR5D5BM14,6RBR
3RBD4DBL4UBD3LBR4RBD2LBL3LBD2RB
R3RBD2LBL3LBD2RBR3RBD2LBL3L":PA
INT(26,15),3
90 DRAW"BM2,21C4UBR4ND3BR4D":GE
T(0,0)-(37,23),H,G
100 PCLS:DRAW"BM7,0C4L6BD2ERFRE
RBD2L7DR7DL5GNR3DNR3FNR4DNR4GNR
3DNR3FNR4DR2GL3FNR6FR3FL4GNRDR5
DL3"
110 GET(0,0)-(7,2),T,G:GET(0,3)
-(7,10),D,G:GET(0,11)-(7,20),B,
G
120 PRINT @129,"QUANTOS JOGADOR
ES (1 OU 2) ?";
130 A$=INKEY$:IF A$<"1" OR A$>"
2" THEN 130
140 P=VAL(A$):NO=P:ER=10000:A(0
,0)=2000000:A(0,1)=2000000:A(1,
0)=2000000:A(1,1)=2000000:A(0,3
)=100000:A(1,3)=100000
150 FOR N=1 TO P:PRINT:PRINT:PR
INT" NOME DO JOGADOR";N;:LINEIN
PUT A$(N-1):IF LEN(A$(N-1))>8 T
HEN A$(N-1)=LEFT$(A$(N-1),8)
160 NEXT
200 FOR N=0 TO 29:FOR M=0 TO NO
-1
202 CLS
210 PRINT @3,"mina de ouro";
220 PRINT TAB(15);A$(0);:IF NO=
2 THEN PRINT TAB(24);A$(1)
230 PRINT @32,"SALDO TOTAL";TAB
(14);A(0,0):IF NO=2 THEN PRINT
@55,A(1,0)
240 PRINT @64,"DINHEIRO";TAB(14
);A(0,1):IF NO=2 THEN PRINT @87
,A(1,1)
250 PRINT @96,"OURO kg";TAB(14)
;A(0,2):IF NO=2 THEN PRINT @119
,A(1,2)
260 PRINT @128,"$ P/ CAVAR";TAB
(14);A(0,3):IF NO=2 THEN PRINT
@151,A(1,3)
270 PRINT @160,"NO. DE MINAS";T
AB(14);A(0,4):IF NO=2 THEN PRIN
T @183,A(1,4)
280 PRINT @192,"PROFUNDI// m";T
AB(14);A(0,5):IF NO=2 THEN PRIN
T @215,A(1,5)
300 PRINT @224,"COTACAO DO OURO
 NO MERCADO":PRINT ER;"POR KG D
E OURO "
400 PRINT @330,A$(M)
500 PRINT "1- PESQUISA E DESENV
OLVIMENTO
510 PRINT "2- LEVANTAMNTO GEOLO
GICO"
520 PRINT "3- CAVAR MAIS 200 M"
530 PRINT "4- VENDER OURO NO ME
RCADO"
540 PRINT "5- PASSA A VEZ";
600 A$=INKEY$:IF A$<"1" OR A$>"
5" THEN 600
620 ON VAL(A$) GOSUB 1000,2000,
3000,4000,5000
700 IF A(M,1)<0 THEN 7000
710 ER=ER+RND(1000)-200
720 IF RND(1600)-A(M,2)<0 GOSUB
 900
740 A(M,0)=A(M,1)+A(M,2)*ER
```

```
750 CLS
790 NEXT M,N
810 CLS
820 PRINT @138,"FIM DE JOGO"
830 PRINT @197,"SALDO TOTAL DE"
;A$(0):PRINT TAB(11);A(0,0)
840 IF NO=2 THEN PRINT TAB(5);"
SALDO TOTAL DE";A$(1):PRINT TAB
(11);A(1,0)
850 PRINT @449,"QUALQUER TECLA
PARA RECOMECAR"
860 IF INKEY$="" THEN 860 ELSE
RUN
900 CLS
905 JK=RND(100)+49:IF JK>A(M,2)
 THEN JK=A(M,2)
910 PRINT @10,"R O U B O"
920 PRINT:PRINT"      FORAM ROUB
ADOS";JK;"KG DE":PRINT"      SEU
```



```
OURO":A(M,2)=A(M,2)-JK:A(M,0)=
A(M,0)-JK*ER
930 PLAY"T401CDEFBAGFED"
940 CLS:RETURN
```

Todos esses programas funcionam de maneira similar, seguindo a mesma estrutura básica e a mesma numeração. Do início até a linha 200, contudo, há algumas variações.

Para começar, a linha 10 pergunta qual o número de jogadores, e a linha 20 verifica se a resposta se encontra no intervalo adequado. A linha 30 acerta os valores de **p** e **nop** de acordo com o número escolhido.

Algumas matrizes são dimensionadas na linha 40, juntamente com a cotação do ouro no mercado — **er**. A matriz **a** armazena informações sobre o saldo dos jogadores; a matriz **c**, informações sobre as minas; a matriz **a$** contém os nomes dos jogadores e a **r** é usada para indicar o começo das escavações na mina em questão. A linha 50 estabelece os saldos e as condições iniciais das minas dos dois jogadores. O valor zero é colocado em **r(1)** e **r(2)**, indicando que as escavações não começaram na primeira mina. Outros valores são dados a seguir: **a(1,1)** e **a(2,1)** contêm o saldo de cada jogador; **a(1,2)** e **a(2,2)**, o total em dinheiro que cada jogador possui; **a(1,3)** e **a(2,3)** indicam a quantidade de ouro de cada um; **a(1,4)** e **a(2,4)**, os custos de mineração; **a(1,5)** e **a(2,5)** revelam o número de minas de cada um e **a(1,6)** e **a(2,6)**, a profundidade das minas.

A linha 70 permite a entrada do nome dos jogadores.

A linha 20 dimensiona as matrizes. A linha 90 seleciona a tela de textos de 32

colunas com fundo amarelo, caracteres pretos e bordas azuis. As teclas de função são desligadas.

A tela de 32 colunas foi escolhida porque permite o uso de cores e de blocos gráficos, tornando o programa mais interessante.

A linha 100 faz com que os oito primeiros caracteres — códigos 0 a 7 — se tornem blocos azuis (veja artigo da página 261). A mesma linha ainda faz com que os oito caracteres seguintes — códigos 8 a 15 — se tornem blocos de cor vermelha.

A linha 110 determina que os caracteres numéricos sejam escritos em branco sobre fundo vermelho.

Depois desses preparativos iniciais, a linha 120 pergunta qual o número de jogadores. A linha 130 verifica se a resposta obtida é válida — um ou dois jogadores.

Em seguida, a linha 140 acerta o valor das variáveis **P**, **NO** — as duas iguais ao número de jogadores — e **ER** — cotação do ouro. Além disso, ela estabelece os valores iniciais de vários elementos da matriz **A**: **A(0,0)** e **A(0,1)**, saldo total dos jogadores; **A(0,1)** e **A(1,1)**, quantia em dinheiro que cada um possui; **A(0,2)** e **A(1,2)**, quantidade de ouro; **A(0,3)** e **A(1,3)**, custo da mineração.

As linhas 150 e 160 perguntam o nome dos jogadores.

O jogo do MSX, como já dissemos, utiliza a tela de textos de 32 colunas, com cor de fundo amarelo e caracteres pretos. Os caracteres numéricos aparecem em branco sobre vermelho, enquanto a tela é mantida com fundo azul. Tudo isso é obtido por meio do comando **VPOKE**, que já usamos outras vezes (veja artigo da página 261).

Na versão para Apple e o TK-2000, nosso jogo começa dimensionando algumas variáveis na linha 20.

Em seguida, a linha 120 pergunta quantas pessoas vão participar e a linha 130 verifica a validade da resposta. A linha 140 guarda o número de jogadores nas variáveis **P** e **NO**, que servirão de contadores e sinalizadores em vários pontos do programa, permitindo que os dados de cada jogador sejam colocados no local correto. Essa linha ainda acerta o valor da cotação do ouro — **ER** —, bem como de vários elementos da matriz **A**: **A(0,0)** e **A(1,0)**, saldo total dos jogadores; **A(0,1)** e **A(1,1)**, total em dinheiro que cada um possui; o par seguinte corresponde à quantidade de ouro e o último, ao custo da escavação nas minas.

As linhas 150 e 160 perguntam o nome dos jogadores.

A linha 10 seleciona o modo **PMODE 3** e a tela é limpa. A linha 20 dimensiona várias matrizes.

Como parte do jogo acontece na tela gráfica, em certos momentos é preciso usar **DRAW** para escrever texto. As linhas 40 a 110 contêm a rotina para escrever na tela de alta resolução, publicada em INPUT anteriormente (veja artigo da página 232).

As linhas 120 e 130 perguntam o número de jogadores e verificam se a resposta é válida. A linha 140 acerta os valores de **P** e **NO** de acordo com o número escolhido. Em seguida, alguns dos elementos da matriz **A** têm seus valores iniciais estabelecidos, a saber: **A(0,0)** e **A(1,0)** contêm o saldo total de cada jogador; o próximo par de elementos contém os saldos em dinheiro; o seguinte, a quantidade de ouro que cada um possui e o último corresponde ao custo da mineração.

As linhas 150 e 160 pedem os nomes dos jogadores.

Daí em diante os programas são bem parecidos. Todos eles têm dois laços **FOR...NEXT** que começam na linha 200 e terminam nas linhas 790 e 800. Esses laços são responsáveis pela exibição do menu e da situação das companhias de mineração na tela.

A variável **N** (**n**, no caso do Spectrum) conta o número de rodadas já jogadas. A variável **NO** (**nop**, no Spectrum) garante a oferta de trinta rodadas a cada jogador. Observe que, mais adiante na listagem, esta mesma variável fará com que a situação das companhias de mineração seja exibida.

A linha 202 escolhe as cores da tela no Spectrum. Nos demais, ela apenas limpa a tela — note que no MSX isto é feito por uma sub-rotina que enche a tela com o caractere de código 0, cuja cor azul foi definida pela linha 110. A linha 210 imprime o título: MINA DE OURO. A linha 220 imprime o nome dos jogadores. O segundo nome só é impresso se a opção para dois jogadores foi escolhida no início.

As linhas 230 a 300 mostram os valores: SALDO TOTAL, saldo em DINHEIRO, saldo em OURO, custo para CAVAR, Nº DE MINAS, PROFUNDIDADE da mina e COTAÇÃO DO OURO NO MERCADO. Se duas pessoas estiverem jogando, os dois valores serão impressos no local correspondente. O programa faz isso com base nos valores das variáveis **nop** ou **NO**.

A linha 400 mostra o nome da pessoa que está jogando no momento. As linhas 500 a 540 oferecem as opções de Pesquisa e Desenvolvimento, Levantamento Geológico, Cavar mais 200 metros, Vender ouro no mercado ou Passar a vez. No Spectrum, a linha 550 solicita ao jogador que faça sua opção. Nos demais micros não há esta linha.

As linhas 600 a 620 usam **INKEY$** ou **GET** para obter a resposta do jogador. Além disso, verificam a validade da resposta e chamam a sub-rotina correspondente à escolha feita.

A linha 700 verifica se o saldo total caiu abaixo de zero, terminando o programa, se for o caso, por meio de uma sub-rotina de finalização. A linha 710 faz com que a cotação do ouro flutue ao acaso; assim, o jogador deverá ter o cuidado de vender o metal nas épocas de "alta".

A linha 720 compara um número aleatório com a quantidade de ouro do jogador, para decidir se haverá um roubo — observe que as chances de roubo aumentam com a quantidade de ouro armazenada. A rotina que cuida do roubo fica nas linhas 900 a 940. A linha 905 decide quanto ouro vai ser roubado, e a linha 920 informa que esta quantidade foi roubada.

A linha 740 calcula o saldo total, adicionando o saldo em dinheiro ao valor do ouro armazenado, calculado com base na cotação atual.

A linha 350 restabelece as cores da tela e a limpa, antes que o **NEXT** envie o programa de volta para a linha 200, para mais uma rodada.

As linhas 810 a 840 contêm a rotina de finalização do jogo, usada quando o saldo de um dos jogadores cai abaixo de zero. Ela mostra na tela as condições finais das companhias de mineração e informa: FIM DE JOGO.

Finalmente, as linhas 850 a 880 permitem uma nova partida.

Na próxima seção, adicionaremos várias sub-rotinas que farão com que o jogo funcione. Haverá rotinas para reduzir os custos de mineração por meio de pesquisa e desenvolvimento, para fazer o levantamento geológico, escavar o solo e vender o ouro no mercado. Além disso, traremos os dados necessários para a elaboração dos gráficos que ilustram as minas e mostram o progresso das escavações.

SPRITES PARA O TRS-80 (3)

- OS CARACTERES DE CONTROLE DO CURSOR
- FIGURAS MAIS COMPLEXAS
- ANIMAÇÃO DE FIGURAS COMPLEXAS

Usando gráficos de baixa resolução, podemos criar figuras semelhantes a sprites. Você aprenderá a armazená-las em uma única variável, mesmo que ocupem várias linhas na tela.

A forma mais avançada de produção de figuras animadas em BASIC, nos micros da linha TRS-80, consiste na concatenação de vários caracteres gráficos e seu armazenamento em uma única variável alfanumérica.

Esse método funciona muito bem, mesmo quando a figura ocupa mais de uma linha na tela. Mas, para que possamos continuar tratando a figura como uma entidade única (por exemplo, para empregar a variável que a armazena dentro de um **PRINT@**), precisaremos incluir na sua definição *caracteres de controle do cursor*.

Já mencionamos, brevemente, que certos caracteres situados na faixa do código ASCII que vai de 0 a 31 servem para o controle de funções do vídeo, quando são utilizados com um **PRINT CHR$**. Por exemplo, **PRINT CHR$(23)** dobra a largura dos caracteres na tela (32 colunas).

Quatro desses caracteres nos interessam na geração de figuras com a técnica que aqui examinaremos. São os caracteres de controle do cursor:

24 recua o cursor uma posição na horizontal
25 avança o cursor uma posição na horizontal
26 desce o cursor uma posição
27 sobe o cursor uma posição

Suponhamos que você queira desenhar na tela um pequeno disco voador, formado por duas linhas. Os códigos da linha de cima serão:

```
100 DATA   174,187,187,187,187,
    187,132
```

e os da linha de baixo:

```
110 DATA   128,143,143,143,143,
    143
```

Para concatenar os códigos em uma única figura, proceda assim: quando o cursor terminar de traçar o último caractere da linha de cima (o 132), faça-o descer uma linha e recuar sete posições (número de bytes da linha de cima do gráfico). Coloque, então:

```
105 DATA   26,24,24,24,24,24,24,
    24
```

Depois, acione as linhas anteriores ao programa que damos a seguir e execute-o para ver o resultado.

```
10 D$=""
20 FOR I=1 TO 21:READ N
30 D$=D$+CHR$(N):NEXT I
40 CLS:PRINT @ 288,D$;
```

A figura aparecerá instantaneamente na tela, com um único **PRINT**!

COMPACTANDO A DEFINIÇÃO

Como a definição da figura inclui muitos códigos idênticos, é possível dar-lhe uma forma mais compacta, recorrendo à função **STRING$** e evitando o laço **FOR...NEXT** e as linhas **DATA**. Digite **NEW** e entre este programa:

```
10 D$     =     CHR$(174)+STRING$
   (5,187)+CHR$(132)+CHR$(26)  +
   STRING$(7,24)+CHR$(128)     +
   STRING$(5,143)
20 CLS:PRINT @256,D$;
```

FIGURAS MAIS COMPLEXAS

O método de concatenação de códigos gráficos em uma variável alfanumérica tem, evidentemente, suas limitações. Se a figura ocupar várias linhas, por exemplo, haverá uma excessiva utilização da memória, já que precisamos de um conjunto de caracteres de controle entre cada linha.

A limitação mais séria, entretanto, está na impossibilidade de se criar uma figura com um número de bytes superior a 255, pois este é o comprimento máximo de uma cadeia de caracteres no BASIC do TRS-80. A melhor alternativa será, então, repartir a figura em vários blocos e usar mais de um **PRINT@** para colocá-la inteira na tela. Ao contrário do que parece, tal procedimento não reduz a velocidade de animação gráfica e os resultados, em geral, são semelhantes aos obtidos com a técnica de movimentação do cursor.

Outra técnica muito útil quando se tem figuras grandes e complexas é a chamada *tríade de posição, comprimento, caractere*. Suponhamos que você queira desenhar um trenzinho, fazendo o pistão da roda se movimentar e nuvens de fumaça escaparem da chaminé. O mais adequado será desenhar primeiro a locomotiva completa, usando a técnica da tríade e, depois, animar as partes que interessam, substituindo sucessivamente duas figuras (por exemplo, o braço do pistão em duas posições).

A tríade consiste em uma seqüência de três números, expressa em uma linha **DATA** de definição da figura. O primeiro número determina a posição @ onde começa a cadeia de caracteres; o segundo define a quantidade de caracteres existentes na cadeia; o terceiro indica o código gráfico dos elementos iguais da cadeia:

```
100 FOR I=1 TO 32:READ N1,N2,N3
110 PRINT @ N1,STRING$(N2,N3);
120 NEXT I
```

ACELERE OS SPRITES

Agora que você já sabe como criar figuras de características semelhantes às dos sprites para os microcomputadores da linha TRS-80, será fácil fazer animação gráfica com eles. É claro que essas figuras não possuem uma propriedade fundamental dos sprites autênticos (como os da linha MSX): a *priorização* de imagens. Explicando melhor, cada sprite recebe um número, que indica ao computador se ele vai obstruir ou ser obstruído por outro sprite que ocupar o mesmo lugar na tela.

De qualquer maneira, é possível elaborar programas de animação gráfica relativamente sofisticados para o TRS-80. Um fator importante na animação é o tempo que leva o computador para traçar um bloco gráfico, definido conforme as técnicas apresentadas nesta série. Se a execução do desenho for muito demorada — como no caso de blocos gráficos grandes ou formados por vários **PRINT @** separados —, a animação gráfica não sairá perfeita.

SIMULAÇÃO: FAÇA A BOLA ROLAR

A matemática fornece vários modelos para o mundo real. Utilizando-os em seus programas, você pode produzir efeitos muito interessantes. Aqui, simulamos a trajetória de uma bola.

Na tentativa de construir um modelo para o mundo que nos cerca, a matemática aplicada e outros ramos da ciência fazem uso de um grande número de equações. Estas resultam do exame de dados gerados por observações e experimentos. A partir desse exame, calcula-se uma equação que explica a relação encontrada de maneira que satisfaça os dados obtidos.

Tais equações relacionam-se à mais variada gama de fenômenos físicos e biológicos — desde o crescimento de uma planta até a trajetória de um cometa no espaço. Esse tipo de informação tem diversos usos, sobretudo porque as equações possibilitam prever como as coisas se comportarão em determinadas circunstâncias.

O significado das equações mais complicadas pode não ser claro à primeira vista. Algumas delas só são compreensíveis para os que estão envolvidos num campo específico de pesquisa. Muitas, porém, descrevem fenômenos que fazem parte da experiência de qualquer pessoa. Já vimos um exemplo no programa que usa uma fórmula para estudar o comportamento de um objeto em queda (veja artigo da página 434).

Mas, sejam as equações complicadas ou não, o resultado dos cálculos consiste simplesmente num número — para o tempo ou qualquer outra coisa —, que é mais um modelo para o mundo real. Além da utilidade já mencionada, essas equações têm outro emprego: elas podem ser programadas no seu micro para produzir uma interessante simulação da realidade. Nesse caso, a equação é utilizada para controlar a apresentação de tela. Assim, no exemplo anterior, poderíamos criar uma imagem para mostrar como se comporta um objeto em queda livre.

É possível empregar qualquer equação de movimento para imitar na tela a movimentação real de um objeto. Essa idéia está na base de muitos programas e de curiosas animações.

A melhor maneira de compreender como isso funciona é trabalhar com um exemplo prático — ou seja, tomar uma série de equações que podem ser encontradas em qualquer livro de física e transformá-las em belas simulações na tela. Como exemplo, vamos começar com algo fácil de simular: a trajetória de uma bola numa mesa de bilhar.

O MODELO DA BOLA

Se você bater numa bola com um taco (ou lhe der um chute), ela se moverá em linha reta até encontrar algum obstáculo. Se houver outra força agindo sobre ela — ventos laterais ou gravidade, digamos —, a trajetória e a velocidade serão modificadas. Em uma boa mesa de bilhar, pode-se desconsiderar a ação de ventos, restando apenas uma força com que se preocupar: o efeito da fricção ou atrito com a mesa, que reduzirá gradualmente a velocidade da bola. Mas, quanto? Aí entra a primeira de nossas equações.

Grosso modo, quanto maior a força que atingir a bola no momento da tacada, mais longe ela irá. Existe uma

- O MODELO DA BOLA
- COMO FAZER A BOLA ROLAR
- UMA MIRA MÓVEL
- COMO A EQUAÇÃO FUNCIONA
- TACADAS MAIS FORTES
- CÁLCULO DAS COORDENADAS
- RAÍZES QUADRADAS
- COMO DESACELERAR
- COMO CALCULAR A VELOCIDADE DA BOLA

equação que calcula a distância a ser percorrida por um objeto, dada a força inicial e alguns outros elementos, como o efeito do atrito. Essa equação tem variadas aplicações em programação. Para vê-la funcionando, digite e execute este programa:

```
10 CLS
20 PRINT "1) Bola de futebol
/ grama longa"
30 PRINT '"2) Bola de golfe /
 grama curta"
40 PRINT '"3) Bola de golfe n
a Lua"
50 PRINT '"4) Bola em mesa de
 bilhar"
60 INPUT "Faca a sua opcao",o
pt
70 IF opt<1 OR opt>4 THEN
GOTO 60
 80 IF opt=1 THEN  LET fg=4
 90 IF opt=2 THEN  LET fg=1.4
 95 IF opt=3 THEN  LET fg=.1
100 IF opt=4 THEN  LET fg=.7
110 CLS
115 PRINT "A velocidade pode s
er de 1 a 15 metros por segund
o"
120 INPUT "Qual a velocidade i
nicial (m/s)",v
125 IF v<1 OR v>15 THEN  GOTO
 120
160 LET dist=(v*v)/(2*fg)
170 CLS
210 PRINT AT 2,4;"Sua bola alc
ancaria"
220 PRINT AT 7,8;dist;"metros"
250 PRINT AT 18,1; FLASH 1;"Qu
alquer tecla para continuar"
260 PAUSE 0
270 GOTO 10
```

```
10 CLS
20 PRINT @96," 1)BOLA DE FUTEBO
L / GRAMA LONGA"
30 PRINT @160," 2)BOLA DE FUTEB
OL / GRAMA CURTA"
40 PRINT @224," 3)BOLA DE GOLFE
 NA LUA"
50 PRINT @288," 4)BOLA EM MESA
DE BILHAR"
60 PRINT:PRINT" QUAL SUA OPCAO
?"
70 K$=INKEY$:IF K$<"1" OR K$>"4
" THEN 70
80 OP=VAL(K$)
90 IF OP=1 THEN FG=4
100 IF OP=2 THEN FG=1.4
110 IF OP=3 THEN FG=.1
120 IF OP=4 THEN FG=.7
130 CLS
140 PRINT" A VELOCIDADE PODE SE
R DE 1 A 15 METROS POR SEGUNDO"
150 PRINT:INPUT" QUAL A VELOCID
ADE INICIAL QUE   VOCE DESEJA D
AR (M/S)";VE
160 IF VE<1 OR VE>15 THEN 130
170 DI=(VE*VE)/(2*FG)
180 CLS
190 PRINT"SUA BOLA ALCANCARIA",
DI;"METROS"
200 PRINT @449,"QUALQUER TECLA
PARA RECOMECAR"
210 IF INKEY$="" THEN 210
220 GOTO 10
```

Para rodar o programa no TRS-80, modifique os números utilizados nas instruções **PRINT@** para o dobro dos que estão na listagem.

```
10 CLS:COLOR 15,4,4:DEFSNG D
20 LOCATE 5,5:PRINT"1) Bola de
futebol em grama longa"
```

```
30 LOCATE 5,7:PRINT"2) Bola de
golfe em grama curta"
40 LOCATE 5,9:PRINT"3) Bola de
golfe na lua"
50 LOCATE 5,11:PRINT"4) Bola em
 mesa de bilhar"
60 LOCATE 7,18:PRINT"Escolha su
a opção:";
70 K$=INKEY$:IF K$<"1" OR K$>"4
" THEN 70
80 OP=VAL(K$)
90 IF OP=1 THEN FG=4
100 IF OP=2 THEN FG=1.4
110 IF OP=3 THEN FG=.1
120 IF OP=4 THEN FG=.7
130 CLS
140 PRINT:PRINT"  A velocidade
em metros por segundo  pode ser
 de 1 a 15."
150 PRINT:INPUT"  Qual velocida
de inicial (em m/s)";VE
160 IF VE<1 OR VE>15 THEN 130
170 DI=(VE*VE)/(2*FG)
180 CLS
190 PRINT:PRINT"  Sua bola perc
orreu ";DI;"metros"
200 LOCATE 0,12:PRINT"Pressione
 qualquer tecla para recomeçar"
210 IF INKEY$="" THEN 210
220 GOTO 10
```

```
10  HOME
20  HTAB 5: VTAB 5: PRINT "1) B
OLA DE FUTEBOL EM GRAMA LONGA"
30  HTAB 5: VTAB 7: PRINT "2) B
OLA DE GOLFE EM GRAMA CURTA"
40  HTAB 5: VTAB 9: PRINT "3) B
OLA DE GOLFE NA LUA"
50  HTAB 5: VTAB 11: PRINT "4)
BOLA EM MESA DE BILHAR"
60  HTAB 8: VTAB 15: PRINT "ESC
OLHA SUA OPCAO ";
70  GET K$: IF K$ < "1" OR K$ >
```

```
 "4" THEN 70
80 OP =  VAL (K$)
90  IF OP = 1 THEN FG = 4
100  IF OP = 2 THEN FG = 1.4
110  IF OP = 3 THEN FG = .1
120  IF OP = 4 THEN FG = .7
130  HOME
140  PRINT : PRINT "  A VELOCID
ADE EM METROS POR SEGUNDO PO-DE
  SER DE 1 A 15."
150  PRINT : INPUT "  QUAL A VE
LOCIDADE INICIAL (EM M/S) ";VE
160  IF VE < 1 OR VE > 15 THEN
130
170 DI = (VE * VE) / (2 * FG)
180  HOME
190  PRINT : PRINT "  SUA BOLA
PERCORREU ";DI;" METROS"
200  VTAB 12: PRINT "PRESSIONE
QUALQUER TECLA PARA RECOMECAR";

210  GET K$
220  GOTO 10
```

O programa começa perguntando a respeito das condições em que se encontra nossa bola hipotética. Em seguida, calcula a distância que ela percorreria, para qualquer velocidade inicial que você escolher. Vários fatores — como atrito, massa e gravidade — são levados em conta.

A equação usada é a seguinte:

$$\text{Distância} = \frac{v\uparrow 2}{2*fg}$$

O **v** corresponde à velocidade inicial e **fg** ao valor que engloba a gravidade e o atrito. Obviamente, o valor de **fg** é diferente para cada uma das opções do programa.

OS RESULTADOS

A equação é solucionada na linha 160 do programa do Spectrum ou 170 do dos outros micros. Você pode ver que, embora na equação haja uma potenciação, não há sinal dessa operação (↑) na linha referida. Ocorre que os computadores calculam mais rapidamente somas e multiplicações do que potências. Assim, quando o número está elevado ao quadrado (nosso caso), é melhor multiplicá-lo por ele mesmo (v*v) do que elevá-lo à segunda potência (v ↑ 2).

A opção 3 — bola de golfe na Lua — fornece resultados muito maiores que as demais porque o seu valor para **fg** é bem menor que o das outras opções, uma vez que a gravidade na Lua corresponde a somente um sexto da gravidade na Terra.

A última opção — a da bola na mesa de bilhar, com a qual trabalharemos a seguir — também oferece resultados altos, porque o atrito é muito pequeno em relação ao da grama.

Os valores do programa para o atrito e a gravidade são apenas aproximados, não exatos. Porém, dão uma idéia razoável de como uma equação desse tipo pode ser usada dentro de um programa para prever a distância percorrida por um objeto sob diversas circunstâncias. Mas ainda não pudemos visualizar o objeto em movimento.

A BOLA EM AÇÃO

A equação apresentada calcula a distância percorrida por um objeto, mas assume que não há nada no caminho dele. A opção da bola de bilhar, no programa dado, fornece um bom exemplo: o resultado pode ser uma distância de 10 metros. Mas mesas de bilhar não têm esse tamanho: a bola rebateria nas bordas da mesa.

Ao contrário do comportamento imprevisível de uma bola de futebol sobre a grama, uma bola de bilhar reage de maneira muito regular quando bate contra a borda da mesa (a não ser que tenha tomado um efeito giratório). Ela rebate espalhando o ângulo de incidência (o ângulo com que a bola atinge a borda). Observando a figura da página 673, você entenderá melhor o que ocorre. A bola saiu do ponto **a** e moveu-se pela reta tracejada até parar no ponto **b**. Como você pode ver, os ângulos formados pela linha da trajetória da bola e uma reta perpendicular à borda da mesa são simétricos.

Mais uma vez, a matemática aplicada fornece uma equação para demonstrar este princípio. Com isso, podemos fazer um programa que simule a trajetória da bola na mesa rebatendo em uma ou mais bordas.

O quadro da última página deste artigo mostra as equações que calculam as posições da bola após bater contra qualquer uma das bordas da mesa. Embora pareça complicado, seu computador faz os cálculos muito rapidamente.

```
10 BORDER 4: PAPER 4: INK 0:
CLS
60 LET bx=128: LET by=76: LET
cx=100: LET cy=112: LET nx=
100: LET ny=112
90 FOR n=6 TO 20: PRINT
PAPER 7;AT n,2;"
                ": NEXT n
100 CIRCLE  OVER 1;bx,by,4
110 PLOT  OVER 1;cx-4,cy: DRAW
 OVER 1;8,0: PLOT  OVER 1;cx,
cy-4: DRAW  OVER 1;0,8
```

Os ângulos formados pela trajetória da bola de bilhar e uma reta perpendicular à borda da mesa são simétricos.

```
120 LET a$=INKEY$: IF a$=""
THEN  GOTO 120
130 IF INKEY$="a" AND cy>12
THEN  LET ny=cy-2: GOTO 190
140 IF INKEY$="q" AND cy<122
THEN  LET ny=cy+2: GOTO 190
150 IF INKEY$="p" AND cx<236
THEN  LET nx=cx+2: GOTO 190
160 IF INKEY$="o" AND cx>20
THEN  LET nx=cx-2: GOTO 190
170 IF INKEY$=CHR$ 32 THEN
GOSUB 500: GOTO 300
190 PLOT  OVER 1;cx-4,cy: DRAW
 OVER 1;8,0: PLOT  OVER 1;cx,
cy-4: DRAW  OVER 1;0,8
200 LET cy=ny: LET cx=nx
210 GOTO 110
300 IF INKEY$="" THEN  GOTO
 300
305 CLS
310 PRINT AT 2,5;"ALCANCE=";
INT (p*p*100/(18*18*.4))/100
320 PRINT AT 5,5;"VELOCIDADE I
NICIAL=";INT (ABS (p*100/18))/
100
330 PRINT AT 10,8;"Outra vez ?
(s/n)"
340 LET a$=INKEY$: IF a$<>"s"
AND a$<>"n" THEN  GOTO 340
350 IF a$="s" THEN  CLS : GOTO
90
360 PAPER 7: CLS : STOP
500 IF cx=bx AND cy=by THEN
LET p=0: RETURN
510 LET p=0
515 PRINT AT 0,0;"
                 "
520 LET p=p+2: IF p=256 THEN
GOTO 510
530 PLOT p-1,168: DRAW 0,7:
PLOT p,168: DRAW 0,7
540 IF INKEY$=CHR$ 32 THEN
GOTO 520
550 CIRCLE  OVER 1;bx,by,4
560 LET dx=cx-bx: LET dy=cy-by
570 LET v=p/18: LET sq=SQR (dx
```

```
*dx+dy*dy)
590 LET t=1
600 PLOT bx-.5,by+.5
610 LET vx=v*dx/sq: LET vy=v*
dy/sq
620 LET bx=bx+vx*t-SGN vx*.1*t
*t: LET by=by+vy*t-SGN vy*.1*t
*t
630 LET v=v-.1
640 IF ABS v<.1 THEN  GOTO 700
650 IF bx<15 THEN  LET dx=-dx:
LET bx=30-bx
660 IF bx>239 THEN  LET dx=-dx
: LET bx=478-bx
670 IF by<8 THEN  LET dy=-dy:
```

Como variar a velocidade de simulação do movimento?

Em primeiro lugar, é necessário estabelecer a distinção entre velocidade simulada de deslocamento da bola (ou de um corpo qualquer), medida em cm/s ou outra unidade de velocidade, e velocidade real de deslocamento da imagem da bola na tela.

A velocidade real com que a bola se deslocará na tela depende de muitos fatores. Um deles é o tempo que o interpretador BASIC leva para resolver todas as equações matemáticas que regem o modelo cinemático. Esse tempo varia conforme o computador e a eficiência do programa. Um micro da linha MSX, por exemplo, é bem mais rápido para resolver equações do que um ZX-81. Este divide o tempo total de execução entre UCP e manutenção da tela, o que retarda o cálculo, pois torna necessário o uso de **SLOW** para que a trajetória da bola seja exibida na tela.

A eficiência do programa também é importante. Um número excessivo de substituições nas equações retarda a solução. Podemos, assim, aumentar a eficiência, usando alguns truques, como a inclusão de poucas equações ou a colocação de várias delas em uma única linha de comando.

É importante, ainda, considerar que o incremento de distância nas direções X e Y (**DX** e **DY**) afeta o número de pontos de trajetória calculados por segundo. Quanto menores esses valores, mais fiel será a simulação e mais precisos serão os resultados dos cálculos. Infelizmente, a velocidade diminuirá. Usar valores muito altos de **DX** e **DY**, por outro lado, é perigoso, pois a simulação pode ficar pouco realista.

```
LET by=16-by
680 IF by>127 THEN  LET dy=-dy
: LET by=254-by
690 GOTO 600
710 IF bx<19 THEN  LET bx=20
720 IF bx<235 THEN  LET bx=235
730 IF by<12 THEN  LET by=12
740 IF by>123 THEN  LET by=123
750 CIRCLE bx,by,4: RETURN
```

```
10 PMODE 3,1:PCLS:DIM B(2),BL(2
),C(0)
30 DRAW"BM5,0C3FRFD6GLGHL2HU6E"
:PAINT(5,5),3
40 DRAW"BM20,0C2D2NLR2DL3FD"
50 GET(0,0)-(9,10),B,G
60 GET(18,0)-(23,5),C,G
70 PCLS 4:SCREEN 1,0
80 LINE(18,58)-(230,170),PRESET
,BF
90 BX=123:BY=110:CX=125:CY=112:
NX=125:NY=112
100 PUT(BX,BY)-(BX+9,BY+10),B,O
R
110 PUT(CX,CY)-(CX+5,CY+5),C,OR
120 NX=CX:NY=CY
130 IF PEEK(341)=247 THEN NY=NY
-2:GOTO 190
140 IF PEEK(342)=247 THEN NY=NY
+2:GOTO 190
150 IF PEEK(343)=247 THEN NX=NX
-2:GOTO 190
160 IF PEEK(344)=247 THEN NX=NX
+2:GOTO 190
170 IF PEEK(345)=247 GOSUB 500:
GOTO 220
180 GOTO 100
190 IF NX<18 OR NX>225 OR NY<58
 OR NY>165 THEN 120
200 PUT(CX,CY)-(CX+5,CY+5),BL,P
SET
210 CX=NX:CY=NY:GOTO 100
220 PUT(BX,BY)-(BX+9,BY+10),B,O
R:PUT(CX,CY)-(CX+5,CY+5),C,OR
230 A$=INKEY$
240 IF INKEY$="" THEN 240
250 CLS:PRINT @33,"ALCANCE=";IN
T(P*P*100/(18*18*.4))/100
260 PRINT @97,"VELOCIDADE INICI
AL=";INT(ABS(P*100/18))/100
270 PRINT @225,"QUER RECOMECAR
(S/N) ?"
280 A$=INKEY$:IF A$<>"S" AND A$
<>"N" THEN 280
290 IF A$="S" THEN SCREEN 1,0:G
OTO 100
300 CLS:END
500 IF CX=BX+2 AND CY=BY+2 THEN
 RETURN
510 P=1
520 P=255 AND (P+2):POKE 345,25
5
530 LINE(P,0)-(P+2,6),PSET,BF:L
INE(P+4,0)-(255,6),PRESET,BF
540 SOUND P,1:IF PEEK(345)=247
THEN 520
550 LINE(18,58)-(230,170),PRESE
T,BF
560 DX=CX-BX-2:DY=CY-3-BY
570 V=P/18:SQ=SQR(DX*DX+DY*DY)
580 BX=BX+4:BY=BY+5:T=1
600 PSET(BX+.5,BY+.5,3)
610 VX=V*DX/SQ:VY=V*DY/SQ
620 BX=BX+VX*T-SGN(VX)*.1*T*T:B
Y=BY+VY*T-SGN(VY)*.1*T*T
630 V=V-.1:IF ABS(V)<.1 THEN 70
0
650 IF BX<18 THEN DX=-DX:BX=36-
BX:SOUND 5,1
660 IF BX>230 THEN DX=-DX:BX=46
0-BX:SOUND 5,1
670 IF BY<58 THEN DY=-DY:BY=116
-BY:SOUND 5,1
680 IF BY>170 THEN DY=-DY:BY=34
0-BY:SOUND 5,1
690 GOTO 600
700 BX=BX-4:BY=BY-5
710 IF BX<18 THEN BX=18
720 IF BX>221 THEN BX=221
730 IF BY<58 THEN BY=58
740 IF BY>160 THEN BY=160
750 RETURN
```

```
10 CLS:COLOR 15,15,15:SCREEN 2,
0
15 BX=123:BY=100:CX=123:CY=100:
NX=CX:NY=CY
20 RESTORE:FOR I=1 TO 8
30 READ A:S1$=S1$+CHR$(A):NEXT
40 FOR I=1 TO 8
50 READ B:S2$=S2$+CHR$(B):NEXT
60 SPRITE$(0)=S1$
70 SPRITE$(1)=S2$
80 LINE(18,38)-(230,170),12,BF
100 PUTSPRITE0,(BX,BY),8
110 PUTSPRITE1,(CX,CY),15
120 K$=INKEY$:IF K$=""THEN 120
130 IF K$=CHR$(28) THEN NX=NX+2
:GOTO 190
140 IF K$=CHR$(29) THEN NX=NX-2
:GOTO 190
150 IF K$=CHR$(30) THEN NY=NY-2
:GOTO 190
160 IF K$=CHR$(31) THEN NY=NY+2
:GOTO 190
170 IF K$=CHR$(32) THEN GOSUB 5
00:GOTO 220
180 GOTO 120
190 IF NX<18OR NX>225 OR NY<38
OR NY>165 THEN NX=CX:NY=CY:GOTO
 120
200 CX=NX:CY=NY:GOTO 100
220 K$=INKEY$:IF K$=""THEN220
230 COLOR 15,4,4:SCREEN0:LOCATE
 5,5:PRINT"Distância percorrida
: ";INT(P*P*100/(18*18*.4))/100
240 LOCATE 5,8:PRINT"Velocidade
 inicial: ";INT(ABS(P*100/18))/
100
250 LOCATE 5,11:PRINT"Outra vez
? (S/N)"
260 K$=INKEY$:IF K$<>"S" AND K$
<>"N" THEN 260
270 IF K$="S" THEN COLOR15,15,1
5:SCREEN 2,0:GOTO 20  ELSE CLS:
END
500 IF CX=BX AND CY=BY THEN RET
URN
510 P=1
520 P=255AND(P+2)
530 LINE(P,0)-(P+2,6),6,BF:LINE
```

```
(P+4,0)-(255,6),15,BF
540 IF INKEY$<>CHR$(32) THEN 52
0
560 DX=CX-BX:DY=CY-BY
570 V=P/18:SQ=SQR(DX*DX+DY*DY)
580 T=1
600 PSET(BX+4.5,BY+5.5),15
605 PUTSPRITE0,(BX,BY),8
610 VX=V*DX/SQ:VY=V*DY/SQ
620 BX=BX+VX*T-SGN(VX)*.1*T*T:B
Y=BY+VY*T-SGN(VY)*.1*T*T
630 V=V-.1:IF ABS(V)<.1THEN700
650 IFBX<18THENDX=-DX:BX=42-BX
660 IFBX>222THENDX=-DX:BX=444-B
X
670 IFBY<38THENDY=-DY:BY=76-BY
680 IFBY>162THENDY=-DY:BY=324-B
Y
690 GOTO 600
700 RETURN
1000 DATA 0,28,62,127,127,127,6
2,28
1010 DATA 0,0,8,8,62,8,8,0
```

```
10  HOME : HGR : HCOLOR= 1: HPL
OT 0,0: CALL 62454
30  FOR I = 1 TO 65
40  READ A
50  POKE 767 + I,A
60  NEXT : POKE 232,00: POKE 23
3,03
80 BX = 123:BY = 110:CX = BX:CY
 = BY:NX = CX:NY = CY:P = 1
90  HCOLOR= 0: FOR I = 30 TO 15
0: HPLOT 30,I TO 250,I: NEXT :
HCOLOR= 3
100  SCALE= 1: ROT= 0: DRAW 2 A
T CX + 4,CY + 4
110  DRAW 1 AT BX,BY
120  GET A$
130  IF A$ = "K" THEN NX = NX +
 2: GOTO 190
140  IF A$ = "J" THEN NX = NX -
 2: GOTO 190
150  IF A$ = "I" THEN NY = NY -
 2: GOTO 190
160  IF A$ = "M" THEN NY = NY +
 2: GOTO 190
170  IF A$ =  CHR$ (32) THEN  G
OSUB 500: GOTO 220
180  GOTO 120
190  IF NX < 30 OR NX > 250 OR
NY < 30 OR NY > 150 THEN NX = C
X:NY = CY: GOTO 120
200  GOSUB 750
210 CX = NX:CY = NY: GOTO 100
220  VTAB 21: PRINT "DISTANCIA
PERCORRIDA: "; INT (P * P * 100
 / (18 * 18 * .4)) / 100
230  PRINT "VELOCIDADE INICIAL:
 "; INT ( ABS (P * 100 / 18)) /
 100
240  PRINT : PRINT "MAIS UMA VE
Z? ";
250  GET A$: IF A$ <  > "S" AND
 A$ <  > "N" THEN 250
260  IF A$ = "S" THEN  HOME : G
OTO 90
270  TEXT : HOME : END
500  IF CX = BX AND CY = BY THE
N  RETURN
510 K = 2: HCOLOR= K
520 P = P + K: IF P = 255 THEN
K =  - 2: HCOLOR= 1
525  IF P = 1 THEN 510
530  HPLOT P - 1,0 TO P - 1,8:
HPLOT P,0 TO P,8
540  GET A$: IF A$ =  CHR$ (32)
 THEN 520
560 DX = CX - BX:DY = CY - BY
570 V = P / 18:SQ =  SQR (DX *
DX + DY * DY)
580 T = 1
600  DRAW 2 AT CX + 4,CY + 4: D
RAW 1 AT BX,BY
610 VX = V * DX / SQ:VY = V * D
Y / SQ
615  HPLOT BX + 4 - ( SGN (VX)
* 6),BY + 4 - ( SGN (VY) * 6)
620 MX = BX + VX * T -  SGN (VX
) * .1 * T * T:MY = BY + VY * T
 -  SGN (VY) * .1 * T * T:
630 V = V - .1: IF  ABS (V) < .
```

```
1 THEN 700
650  IF MX < 30 THEN DX =  - DX
:MX = 64 - MX
660  IF MX > 240 THEN DX =  - D
X:MX = 480 - MX
670  IF MY < 30 THEN DY =  - DY
:MY = 64 - MY
680  IF MY > 140 THEN DY =  - D
Y:MY = 280 - MY
690  GOSUB 750:BX = MX:BY = MY:
 GOTO 600
700  RETURN
750  HCOLOR= 0: DRAW 1 AT BX,BY
: DRAW 2 AT CX + 4,CY + 4: HCOL
OR= 3: RETURN
1000  DATA  2,0,6,0,45,0,9,45,4
5,21,63,63,63,23,45,45,45,45,62
,63,63,63,55,45,45,45,45,62,63,
63,63,55,45,45,45,45,30,63,63,6
3,14,45,45,150,0
1010  DATA  36,36,55,54,62,63,5
5,45,45,45,45,37,63,191,50,54,3
9,36,4,0
```

Para o TK-2000, faça as seguintes modificações no programa anterior:

```
10  HOME : HGR : HCOLOR= 1
15  FOR I = 1 TO 160: HPLOT 1,I
 TO 279,I: NEXT
```

O programa desenha um retângulo no meio da tela: esta é a área (ou a mesa) na qual o objeto pode se mover.

UMA MIRA MÓVEL

No meio do retângulo há uma bola e, também, um cursor que pode estar escondido atrás dela. No Spectrum, pressione a tecla Q para movê-lo para cima, A para baixo, O para a esquerda e P para a direita; no TRS-Color e no MSX, use as setas para a movimentação; no Apple e no TK-2000, tecle I para cima, M para baixo, J para a esquerda e K para a direita. O cursor é representado por uma cruz, que se move por toda a área disponível para indicar a direção de movimento da bola. Quando tiver posicionando o cursor no lugar desejado, pressione a barra de espaços (ou a tecla **SPACE**, no Spectrum).

Uma linha horizontal surgirá no topo da tela, aumentando à medida que se mantém a barra de espaços apertada. O tamanho da linha mostra a força com que a bola será impulsionada: quanto maior a linha, maior a força.

Assim que se solta a barra de espaços, o computador limpa o retângulo e desenha uma série de pontos que mostram a trajetória da bola. Os pontos ficam mais próximos quando a bola perde velocidade. No Apple, para colocar a bola em movimento, é necessário pressionar uma tecla qualquer após ter definido a força.

Tente variar a posição do cursor e usar diferentes velocidades na bola, para verificar o efeito.

COMO FUNCIONA

Aqui, os princípios da reflexão estão muito claros. Toda vez que a bola bate contra uma borda, ela rebate no mesmo ângulo com que chegou. Isso acontece em qualquer lado da mesa.

O programa começa com a inicialização de algumas variáveis e com a escolha do modo gráfico adequado e das cores. O do TRS-Color também dimensiona três vetores, define dois UDG (Caracteres Definidos pelo Usuário) e os coloca nos vetores (um para a bola, um para a cruz e outro em branco). Nos programas do Apple e do MSX, usam-se duas formas predefinidas, uma para a bola e outra para a cruz. No MSX, isto é feito com Sprites, que são muito fáceis de definir e têm a vantagem de não apagar outros desenhos da tela que estejam em plano diferente. No Apple, utiliza-se uma tabela de formas. Não é tarefa simples montar uma delas sem o auxílio de um editor, mas o efeito final compensa. Para animar a figura, será preciso ter alguns cuidados (como apagar o desenho anterior).

As variáveis definidas são **BX**, **BY** (coordenadas **X** e **Y** para a bola); **CX**, **CY** (coordenadas para a cruz) e **NX** e **NY** (novas coordenadas para a cruz). **NX** e **NY** são as variáveis usadas para mover a cruz pela tela.

Depois desses procedimentos iniciais, um retângulo é desenhado na tela e a rotina **INKEY$** ou **GET**, que lhe permite mover o cursor, entra em ação. A rotina do TRS-Color usa **PEEK** nas posições de teclado, para possibilitar a auto-repetição das teclas.

Além de verificar a direção em que o cursor se move, a rotina checa também se alguma tecla está sendo pressionada. Caso haja pressão sobre a barra de espaços, o computador vai para a sub-rotina 500, que verifica a posição do cursor: se ele está sobre a bola, a força é imediatamente zerada e o computador retorna da sub-rotina. O TRS-Color não precisa especificar **P** (a variável para força) como 0, já que seu BASIC assume este valor para qualquer variável, a não ser que se determine algum outro.

Todo esse procedimento tem a finalidade de evitar a ocorrência de erros, pois, quando o cursor está sobre a bola, o computador não sabe em qual direção deve movê-la.

Se o cursor está numa posição válida, o fator força é definido como 0 (no Spectrum) ou 1 (nos outros micros). Isso ocorre no momento em que a barra horizontal é apagada.

TACADAS MAIS FORTES

A variável **P** é incrementada (tem seu valor aumentado) de 2, cada vez que o computador detecta uma pressão na barra de espaços. Quando atinge o valor 255, **P** volta a ser 0 — e a linha horizontal, que cresce com **P**, é novamente apagada. O computador permanece nesse círculo vicioso até que a barra de espaços se libere. No Apple, em vez da força zerar ao atingir o máximo, ela vai diminuindo aos poucos.

Assim que a condição da linha 540 deixa de ser verdadeira — ou seja, quando a barra de espaços é liberada — o computador continua a execução do programa que, neste ponto, varia um pouco de uma máquina para outra. O Spectrum e o TRS-Color apagam a bola e traçam sua trajetória. O MSX e o Apple mostram a bola em movimento, deixando um rastro em sua trajetória. Mais adiante, explicaremos como funciona esta parte do programa.

Ao terminar os cálculos e o desenho do caminho da bola, o computador retorna para a linha 170. Em seguida, vai para a linha 300, no Spectrum, ou 220 nos outros micros.

O TRS-Color coloca a bola em sua nova posição e a cruz no mesmo lugar onde ela estava. As declarações **OR** que aparecem no fim de cada **PUT** evitam que a bola ou a cruz apaguem alguma coisa (pontos) da tela.

O computador espera que uma tecla seja pressionada para, então, apagar a tela e mostrar a velocidade inicial da bola e a distância percorrida. Em seguida, oferece mais uma tentativa. O Apple, novamente, apresenta uma diferença: a tela não é apagada, pois as informações são mostradas na janela de texto (últimas quatro linhas) da tela de alta resolução.

Como mostra a figura da página 653, para prever a direção que um objeto tomará após bater numa borda, basta estudar o ângulo com que atingiu a parede. Infelizmente, os computadores não dispõem de nenhum recurso para medir ângulos diretamente, precisando recorrer a uma equação.

CÁLCULO DAS COORDENADAS

Cada um dos programas define variáveis para as posições da bola e da cruz (há, de fato, dois conjuntos de coordenadas para a cruz, mas apenas um é usado aqui). O computador necessita de um outro conjunto — que contém a diferença entre as posições da cruz e da bola. De posse desses dados, pode calcular a direção da bola.

As variáveis que indicam a diferença entre as posições são **DX** e **DY**, calculadas na linha 560.

O programa utiliza muitas outras variáveis para os cálculos. O valor de **V**, variável usada para a velocidade inicial, é uma proporção da força que foi escolhida para impulsionar a bola (o valor exato desta proporção varia um pouco de um micro para outro, em razão de diferenças da tela).

RAÍZES QUADRADAS

A variável **SQ** iguala-se à raiz quadrada de **(DX*DX + DY*DY)**. Esta é, na realidade, a distância entre a cruz e a bola, distância que é calculada por meio do teorema de Pitágoras, que estabelece que, para qualquer triângulo retângulo, o quadrado da hipotenusa (como é denominado o lado maior do triângulo) é igual à soma dos quadrados dos catetos (nome que recebem os outros dois lados).

A variável **T** corresponde ao intervalo de tempo usado nas equações. Se seu valor for maior, o programa funcionará mais rapidamente, porém, menos pontos serão traçados: como a razão entre o tempo e a distância fica menor, o número de pontos também diminui. Assim que se determina o intervalo de tempo, o computador começa a traçar os pontos da trajetória.

Em seguida, o programa utiliza mais duas variáveis para os cálculos. São os valores para a velocidade, nos eixos **X** e **Y** (ou horizontal e vertical). Em outras palavras, **VX** contém a velocidade com que a bola se desloca horizontalmente e **VY**, a velocidade de seu deslocamento vertical. As duas variáveis podem contar valores negativos: nesse caso, a bola se desloca da direita para a esquerda (para **VX**) ou de baixo para cima (para **VY**).

O valor dessas duas variáveis é calculado, como se pode verificar na linha 610, multiplicando-se a velocidade (**V**) pela variável **DX** (ou **DY**) e dividindo-se o resultado pela distância entre o centro da bola e o centro da cruz (variável **SQ**).

MICRO DICAS

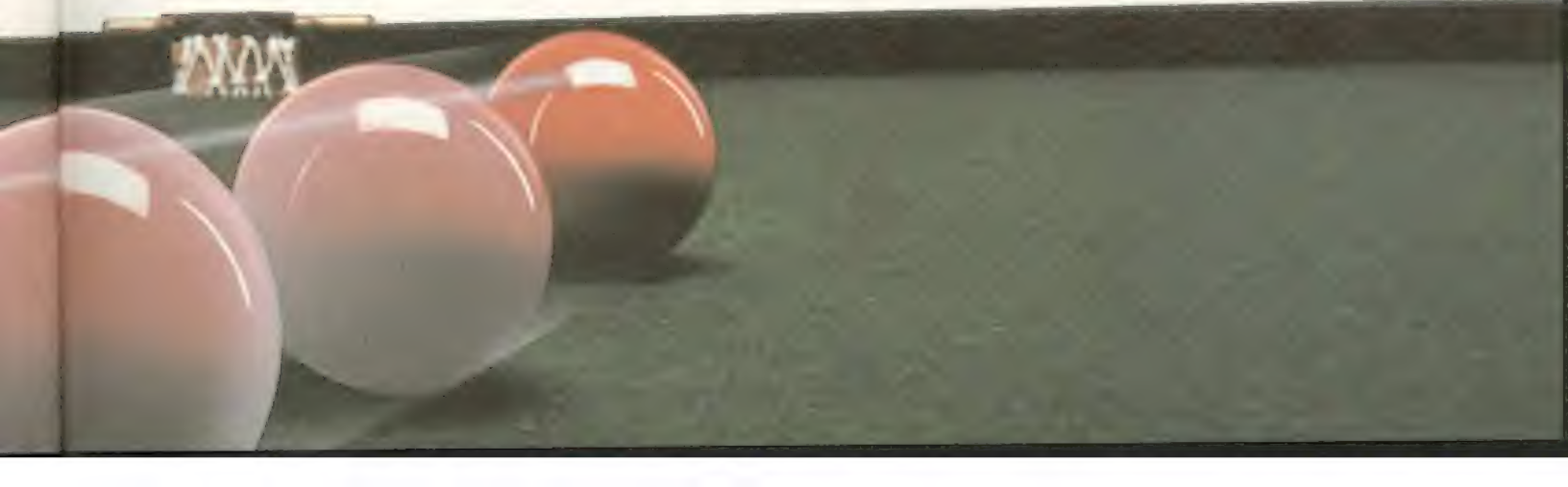
UTILIZAÇÃO DE MODELOS DINÂMICOS

Para quem não gosta muito de matemática ou física, pode parecer que os programas deste artigo não têm muitas aplicações práticas. Na verdade, estas são numerosíssimas.

Qualquer pessoa que tenha a aspiração de se tornar um programador competente — sobretudo nas áreas de jogos e programas educativos — precisa conhecer bem o mecanismo de simulação de modelos dinâmicos.

Veja, abaixo, alguns exemplos do que você pode fazer usando como base os programas deste artigo:

- um programa que simule um jogo de bilhar. Com o joystick ou o teclado, o jogador orienta o ângulo e a força de batida de um taco.
- um jogo de fliperama.
- um programa educativo que mostre como as moléculas se deslocam em um recipiente, colidindo continuamente umas com as outras. Quanto maior a densidade da matéria (por exemplo, densidade de gás), maior o número de colisões. Colisões geram calor; assim, a temperatura do gás pode ser estimada a partir de sua densidade ou pressão (esta é uma das leis fundamentais da física).
- um programa educativo para demonstrar o efeito de diferentes coeficientes de atrito, ou diferentes valores da força de gravidade sobre a movimentação livre de uma bola.

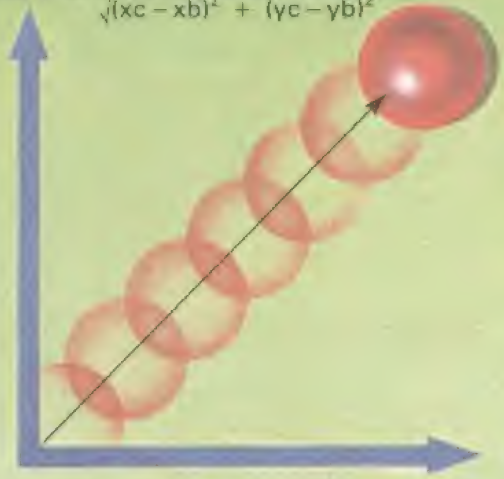


Como calcular a velocidade da bola

O programa recorre a várias equações para calcular a nova posição da bola e sua velocidade. As equações do diagrama separam a velocidade em suas componentes, com a projeção nos eixos **X** e **Y**. O computador resolve essas duas equações para cada posição da trajetória da bola, porém, sob uma forma um pouco diferente da que foi apresentada aqui. As constantes (Xc – Xb) e (yc – Xb) são substituídos no programa por **DX** e **DY**. Com elas, o computador calcula então a nova velocidade a cada ponto, levando em consideração a desaceleração.

Como mostra a fórmula, ele multiplica a diferença entre as coordenadas **X** e **Y** da bola e do cursor pela velocidade real, e divide o resultado pela linha de baixo da razão: a raiz quadrada da diferença entre as coordenadas **X** da bola e do cursor elevada ao quadrado, somada à diferença das coordenadas **Y** da bola e do cursor elevada ao quadrado.

$$\text{vel Y} = \frac{V\ (yc - yb)}{\sqrt{(xc - xb)^2 + (yc - yb)^2}}$$



$$\text{vel X} = \frac{V\ (xc - xb)}{\sqrt{(xc - xb)^2 + (yc - yb)^2}}$$

As duas equações seguintes calculam as novas coordenadas **X** e **Y**. Veja:

```
Xn = Xb + Vx*T-0.01*T
Yn = Yn + Vy*T-0.01*T
```

A nova posição da bola é calculada adicionando-se um número à antiga coordenada **X** e **Y**. Para calcular o número a ser adicionado, multiplica-se a velocidade no eixo **X** ou **Y** pelo tempo e, então, subtrai-se uma pequena fração de **T** ao quadrado, para levar em conta a desaceleração.

A velocidade é diminuída de uma pequena quantidade (0.1) e, antes de se reiniciar o *loop*, marca-se a nova posição da bola.

A NOVA POSIÇÃO DA BOLA

Neste ponto, o seu micro já dispõe de todos os valores necessários para calcular a nova posição da bola. A linha 620 faz esse cálculo para os novos valores de **BX** e **BY**.

O cálculo para cada uma das coordenadas é o seguinte:

```
BX = BX+VX*T-SNG(VX)*.1*T*T
```

Embora pareça complicado, não é. A fórmula adiciona um valor a **BX** (ou **BY**) para encontrar a nova coordenada. O número pode ser negativo, dependendo da direção em que a bola se desloque. Em vários pontos desta parte do programa para o Apple, você irá encontrar as variáveis **MX** e **MY** em lugar das variáveis **BX** e **BY**, porque o computador precisa das coordenadas antigas da bola para poder apagá-la, antes de desenhá-las em sua nova posição.

O primeiro passo é adicionar a **BX** à distância que a bola percorre. Esta distância é avaliada em dois estágios. Inicialmente, multiplica-se a velocidade pelo intervalo de tempo. Lembre-se de que a velocidade nada mais é do que a distância percorrida dividida pelo tempo gasto. Assim, para calcular a distância, basta multiplicar a velocidade pelo tempo.

Infelizmente, o programa precisa fazer um ajuste, pois as velocidades dadas pelas variáveis **VX** e **VY** não são iguais à velocidade real da bola em movimento — cada uma delas corresponde à velocidade de deslocamento em um eixo (horizontal ou vertical). Isso explica o estranho cálculo no fim da fórmula, envolvendo o sinal de **VX** e uma fração de **T** ao quadrado.

A função **SGN** fornece o resultado 1, -1 ou 0, conforme o argumento (número dentro dos parênteses) seja positivo, negativo ou zero. Ela é usada aqui para que se possa, empregando a mesma fórmula, subtrair o valor **.1*T*T** do termo anterior, no caso da velocidade no eixo ser negativa, ou somá-lo, no caso da velocidade ser positiva.

O resultado após cálculos desta linha corresponde à nova coordenada (para o eixo **X** ou **Y**) da bola. O cálculo é feito para os dois eixos.

COMO DESACELERAR

O programa diminui a velocidade de 0.1 e checa se ela é menor que 0.1. Em caso afirmativo, o micro vai para o fim da rotina, que é explicado abaixo. Em caso negativo, quatro linhas são usadas para verificar se alguma das bordas foi atingida no caminho para a nova posição (utilizam-se quatro **IF...THEN** para checar se as novas coordenadas estão caindo fora dos limites da mesa). Se a bola bate contra uma das bordas, o valor da variável **DX** (ou então **DY**, conforme a borda atingida tenha sido a superior ou a inferior) é igualado a menos **DX**.

Os valores de **BX** e **BY** também são alterados, de modo a colocar a bola na posição correta. Os números subtraídos das coordenadas antigas são o dobro dos limites da mesa. Assim, a maior coordenada da mesa no Spectrum é 127, o que significa que, toda vez que a bola encontra a borda superior, **BX** fica 254-**BX** (254 é o dobro de 127).

Depois das checagens, o computador calcula o próximo ponto. O fim da rotina, alcançado quando a velocidade se torna menor que 0.1, verifica se a bola não vai ser desenhada sobre a borda, antes de desenhá-la na tela.

Algumas das coordenadas usadas nesta verificação são diferentes daquelas usadas na anterior, porque o que vai ser desenhado é uma bola e não um ponto — e, é claro, a bola ocupa muito mais espaço. Assim, as coordenadas necessitam de algum ajuste.

SUGESTÕES

Você pode dar vários usos ao programa aqui apresentado. Uma idéia é transformá-lo em um jogo de *snooker*. Se esta for sua opção, bastará acrescentar algumas caçapas, para que as bolas caiam dentro, e uma rotina de contagem de pontos. Outra possibilidade é aproveitar esse programa para um jogo de *squash*, ou algo do tipo.

UM ASSEMBLER PARA O TRS-80

- MODIFICAÇÕES QUE DEVEM SER FEITAS NO PROGRAMA DO MSX
- CARACTERÍSTICAS DOS PROGRAMAS ASSEMBLY
- FUNCIONAMENTO DO ASSEMBLER

Com algumas modificações, o programa Assembler que apresentamos anteriormente para o MSX servirá também para os usuários do TRS-80. Veja aqui as adaptações necessárias.

Traduzir mnemônicos para código de máquina certamente é uma tarefa bastante cansativa. E, além de trabalhoso, colocar os códigos usando o monitor de seu TRS-80 pode dar origem a muitos erros — fatais, quando se trata de linguagem de máquina.

Neste artigo, você verá como utilizar o computador para fazer esse serviço. Com algumas modificações, o programa que publicamos anteriormente para o MSX servirá também para o TRS-80. Ele se destina a sistemas com 48K de memória, sendo que as funções de gravação e leitura dos programas em linguagem Assembly só funcionam em sistemas com disquetes. As outras funções do Assembler funcionarão normalmente em sistemas com fita cassete.

Como o programa foi escrito em BASIC, ele é um tanto lento, levando alguns minutos para montar um programa longo. Contudo, funciona muito bem.

Digite o Assembler para o **MSX** apresentado no artigo da página 401, com as seguintes modificações:

```
5000 CLS:PRINT@468,"Um instante
, por favor":CLEAR 500:DIM K$(1
10),K(110),M(110):H$="012345678
9ABCDEF":G$="0123456789abcdef"
5200 CLS:PRINT@86,"ASSEMBLER":P
RINT:PRINTTAB(16)"(c)  carregar
 do disco":PRINT:PRINTTAB(16)"(
g)  gravar no disco":PRINT:PRIN
TTAB(16)"(e)  editar linha":PRI
NT:PRINTTAB(16)"(m)  montar":PR
INT:PRINTTAB(16)"(a)  apagar li
nha":PRINT:PRINTTAB(16)"(l)  li
star"
5210 A$=INKEY$:IFA$=""THEN5210
5220 JJ=INSTR("cgemal",A$)
5230 IFJJ=0THENPRINT"<"A$"> ???
":FORJ=1TO500:NEXT:GOTO5200
5240 CLS:ONJJGOSUB10420,10450,1
0490,5290,10710,10760
5250 PRINT:PRINT"Qualquer tecla
 para continuar"
5260 A$=INKEY$:IFA$=""THEN5260
5270 GOTO 5200
10420 INPUT"Nome do arquivo ";A
R$
10430 CLS:PRINT@465,"Carregando
 programa fonte"
10440 OPEN"I",1,AR$:INPUT#1,N:F
ORJ=1TON:INPUT#1,T$(J):NEXT:CLO
SE1:RETURN
10450 INPUT"Nome do arquivo ";A
R$
10460 CLS:PRINT@466,"Gravando p
rograma fonte"
10470 OPEN"O",1,AR$:PRINT#1,N:F
ORJ=1TON:PRINT#1,CHR$(34);T$(J)
;CHR$(34):NEXT:CLOSE1:RETURN
10490 PRINT"Qual o numero da li
nha (multiplo de 10) ";
10500 INPUTK:CLS
10510 K2=K/10:IFK2>NTHENK2=N+1:
N=N+1:T$(K2)="":PRINT@960,""
10520 IFK2<.1THENK2=.1
10530 IFK2=INT(K2)THEN10550
10540 K2=INT(K2)+1:FOR K3=NTOK2
-1STEP-1:T$(K3+1)=T$(K3):NEXT:N
=N+1:T$(K2)=""
10550 P1=16326:P0=P1
10560 PRINT@896,K;TAB(6)T$(K2)+
STRING$(20,32):P9=P0+LEN(T$(K2)
)
10570 IFP1<P0THENP1=P0
10580 IFP1>P9THENP1=P1-1
10590 FOR I=-1 TO 1:POKE P1+I,3
2-(I=0)*99:NEXT I
10600 P7=0:A$=INKEY$:IFA$=""THE
N10600
10610 IFA$=CHR$(13)THEN10700
10615 IFA$=CHR$(27)THENRETURN
10620 IFA$=CHR$(9)THENP1=P1+1:G
OTO10580
10630 IFA$=CHR$(8)THENP1=P1-1:G
OTO10570
10640 IFA$=CHR$(10)THENA$="":GO
TO10670
10650 IFA$=CHR$(91)THENA$=" "+M
ID$(T$(K2),P1-P0+1,1):P7=-1:GOT
O10670
10660 IFA$<" "THEN10600
10670 IFP1-P0+1>LEN(T$(K2))THEN
T$(K2)=LEFT$(T$(K2),P1-P0)+A$:G
OTO10690
10680 T$(K2)=LEFT$(T$(K2),P1-P0
)+A$+RIGHT$(T$(K2),LEN(T$(K2))-
P1+P0-1)
10690 P1=P1-(LEN(A$)>0)+P7:GOTO
10560
10700 PRINT@64,"";K=K+10:GOTO 1
0510
10710 IFN=0THENCLS:PRINT"Nada a
 apagar":RETURN
10720 CLS:PRINT"Qual o numero d
a linha (multiplo de 10) ";
10730 INPUTK:K2=K/10
10740 IFK2>NORK2<1ORK2<>INT(K2)
THENPRINT"Esta linha nao existe
":RETURN
10750 K=K2:FORK3=K2TON:T$(K3)=T
$(K3+1):NEXT:N=N-1:PRINT@190,K*
10;"  ";T$(K):RETURN
10760 IFN=0THENPRINT"Nada a lis
tar":RETURN
10770 PRINT"Quais os numeros da
 primeira e da ultima linha (mu
ltiplos de 10)";
10780 INPUTK,K2:K=INT(K):K2=INT
(K2):K1=K/10:K2=K2/10
10790 IFK2>NTHENK2=N
10800 IFK1<1THENK1=1
10810 IFK2<K1ANDK2=NTHENRETURN
10820 IFK2<K1THENCLS:PRINT"Nume
ros de linha invalidos":GOTO 10
770
10830 CLS:PRINT@192,:FORK3=K1TO
K2:PRINTK3*10" "T$(K3):NEXT
10840 RETURN
```

Algumas das linhas do programa — a 5450 e a 5750, por exemplo — são muito longas e podem dar problemas no momento de digitação. Para evitar que isso ocorra, digite as linhas exatamente como estão impressas, sem incluir espaços entre os comandos. Se mesmo assim o computador não aceitar uma linha devido ao seu tamanho, aperte **ENTER**, fazendo o começo da linha ir para a memória. Em seguida, edite a linha usando o comando **X** da edição.

Observe que nas linhas 5030, 5050, 5390 e 5500 encontram-se quatro caracteres em branco entre aspas; nas linhas 5660 e 5770, três.

Antes de montar o programa, proteja o topo da memória do computador para ali colocar os códigos. Faça-o logo que ligar a máquina — ou ativar o BASIC —, respondendo adequadamente à pergunta "Memória usada?".

COMO USAR O PROGRAMA

Ao executar o Assembler, um menu será exibido na tela. Para digitar um programa em Assembly tecle "e". No TRS-80, é necessário pressionar ao mesmo tempo as teclas **SHIFT** e @ (arroba) para usar as letras minúsculas. O computador perguntará, então, o número da linha. Cada instrução deve ficar em uma linha numerada, como acontece também no BASIC. Os números das linhas precisam sempre ser múltiplos de 10 e apenas um mnemônico Assembly com seus respectivos operandos deve ser introduzido em cada uma delas.

A primeira linha de um programa em Assembly especifica a origem ou endereço inicial do programa em código. Isto é feito pelo comando **org**, de modo que a primeira linha de seu programa ficará mais ou menos assim:

```
10 org -10000
```

Se os mnemônicos não forem escritos em letras minúsculas e os números não estiverem todos situados entre −32768 e 32767, haverá erro na execução do Assembler.

Os mnemônicos empregados seguem o padrão Z-80, com exceção de **ret**. Usado sozinho, este comando é aceito normalmente. Porém, nos casos de retorno condicional — **ret nz**, por exemplo — devemos usar **rts** em seu lugar.

Números hexadecimais precisam ser precedidos do sinal **$**; números binários, do sinal **%**. Números que não tenham sinal na frente serão interpretados como decimais.

Qualquer palavra que não corresponda a um mnemônico será interpretada como um rótulo. Evite palavras parecidas ou números.

Programas Assembly devem terminar com **end**, comando que indica o final da listagem.

Para editar uma linha — antes de apertar **ENTER** — utilizam-se as setas do teclado: "direita" e "esquerda" movem o cursor ao longo da linha; "para baixo" é usada para inserir e "para cima", para apagar caracteres. A edição das linhas é bem lenta. Muitas vezes, os caracteres digitados demoram a aparecer no vídeo.

Para sair do modo de edição, pressione simultaneamente as teclas **SHIFT** e "seta para cima". Elas devem substituir o **ENTER**, pois, se o utilizarmos depois que tivermos digitado a última linha, uma linha em branco será incorporada à listagem.

De volta ao menu, pressione "l", para obter uma listagem do programa em Assembly na tela.

Se houve algum erro durante a introdução do programa, volte ao menu e pressione "e"; em seguida, forneça o número da linha que deseja mudar.

Se quiser inserir uma linha entre duas já existentes, digite-a usando um número intermediário, no modo de edição. Quando listar o programa, verá que ele foi renumerado e que a nova linha ocupa o lugar certo. Porém, apenas uma linha pode ser introduzida dessa maneira de cada vez.

Para apagar uma linha, volte ao menu, tecle "a" e forneça o número da linha que quer eliminar.

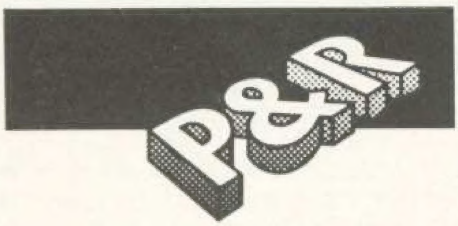
Quando estiver satisfeito com o programa Assembly (*programa-fonte*), volte ao menu e tecle "m" para que ele seja montado na memória. Ao mesmo tempo, o computador fornecerá uma listagem dos códigos hexadecimais equivalentes: o *programa-objeto*.

Uma vez montado o programa, o endereço final da rotina em código aparecerá na tela. Se for detectado algum erro durante a montagem, volte ao menu e edite a linha correspondente, repetindo, então, a montagem.

A opção "g" do menu grava o programa-fonte — os mnemônicos, portanto — em disco. A opção "c" faz a operação inversa.

Para gravar em fita cassete o programa que acaba de ser montado, você deve teclar **RESET**. Contudo, não pressione por muito tempo, nem mais de uma vez, sob pena de perder o programa-objeto e entrar no monitor de linguagem de máquina do TRS-80. Nos sistemas sem disquete não há como gravar o programa-fonte em Assembly.

Para executar o programa em código, use as instruções **DEF USR** e **A = USR(0)** (veja artigo da pág. 88).



O que é um Disassembler?

Como o próprio nome indica, o *Disassembler* executa uma função oposta ao *Assembler* — ou seja, enquanto o Assembler é um programa que traduz códigos mnemônicos (simbólicos) em código de máquina (números binários, diretamente executáveis pela UCP), o Disassembler reconstitui o programa em linguagem Assembly.

Como funciona um programa Disassembler? Da mesma maneira que no programa Assembler listado neste artigo, a base do processo de tradução é uma tabela. Só que o processo de desmontagem (disassembler) é mais complexo do que o de montagem, já que diversas instruções elementares do microprocessador Z-80 requerem dois ou mais bytes para serem totalmente armazenadas (ou seja, código de operação, mais operandos ou argumentos). Como o significado de um determinado byte na memória é diferente conforme o ponto da instrução em que ele se encontra, a ordem de leitura do programa em código de máquina é muito importante para o Disassembler. Uma única falha de interpretação de um byte invalidará toda a desmontagem até o final do programa.

O Disassembler não é um programa fácil de ser desenvolvido, principalmente em BASIC. Mas existem diversos programas Disassembler comercialmente disponíveis para o TRS-80.

UTILIZAÇÃO DAS ROTINAS DA ROM NO SEU PROGRAMA EM ASSEMBLER

O Assembler é mais elementar do que o BASIC e outras linguagens de programação de alto nível. No entanto, é muito difícil utilizá-lo, mesmo para a programação de tarefas simples, pois ele exige que todas as operações da UCP, da memória, e de entrada/saída sejam especificadas da maneira mais minuciosa possível.

Mas existe um recurso capaz de aliviar esse cansativo trabalho: chamar as rotinas prontas que existem na memória ROM do computador.

Os micros da linha TRS-80 dispõem de um sistema operacional elementar e de um interpretador BASIC, ambos gravados na memória ROM. Esses programas contêm centenas de rotinas de operação do computador, que podem ser chamadas pelo comando **JSR** (desvio para sub-rotina), seguido do endereço onde elas começam.

Em geral, cada rotina necessita de um ou mais argumentos de entrada, e devolve um ou mais argumentos de saída. Os registros de intercâmbio de dados variam, mas comumente são os pares de registros internos do Z-80, como o HL e outros. Para utilizar as rotinas da ROM, você precisará, portanto, de um "mapa da mina". Existem vários livros que fornecem todos os endereços e argumentos.

CLASSIFICAÇÃO DOS MICROCOMPUTADORES POR LINHA

LINHA	FABRICANTE	MODELO
Apple II +	Appletronica	Thor 2010
Apple II +	CCE	MC-4000 Exato
Apple II +	CPA	Absolutus
Apple II +	CPA	Polaris
Apple II +	Digitus	DGT-AP
Apple II +	Dismac	D-8100
Apple II +	ENIAC	ENIAC II
Apple II +	Franklin	Franklin
Apple II +	Houston	Houston AP
Apple II +	Magnex	DM II
Apple II +	Maxitronica	MX-2001
Apple II +	Maxitronica	MX-48
Apple II +	Maxitronica	MX-64
Apple II +	Maxitronica	Maxitronic I
Apple II +	Microcraft	Craf II Plus
Apple II +	Milmar	Apple II Plus
Apple II +	Milmar	Apple Master
Apple II +	Milmar	Apple Senior
Apple II +	Omega	MC-400
Apple II +	Polymax	Maxxi
Apple II +	Polymax	Poly Plus
Apple II +	Spectrum	Microengenho I
Apple II +	Spectrum	Spectrum ed
Apple II +	Suporte	Venus II
Apple II +	Sycomig	SIC I
Apple II +	Unitron	AP II
Apple II +	Victor do Brasil	Elppa II Plus
Apple II +	Victor do Brasil	Elppa Jr.
Apple IIe	Microcraft	Craft IIe
Apple IIe	Microdigital	TK-3000 IIe
Apple IIe	Spectrum	Microengenho II
MSX	Gradiente	Expert GPC-1
MSX	Sharp	Hotbit HB-8000
Sinclair Spectrum	Microdigital	TK-90X
Sinclair Spectrum	Timex	Timex 2000
Sinclair ZX-81	Apply	Apply 300
Sinclair ZX-81	Engebras	AS-1000
Sinclair ZX-81	Filcres	NEZ-8000
Sinclair ZX-81	Microdigital	TK-82C
Sinclair ZX-81	Microdigital	TK-83
Sinclair ZX-81	Microdigital	TK-85
Sinclair ZX-81	Prologica	CP-200
Sinclair ZX-81	Ritas	Ringo R-470
Sinclair ZX-81	Timex	Timex 1000
Sinclair ZX-81	Timex	Timex 1500
TRS-80 Mod. I	Dismac	D-8000
TRS-80 Mod. I	Dismac	D-8001/2
TRS-80 Mod. I	LNW	LNW-80
TRS-80 Mod. I	Video Genie	Video Genie I
TRS-80 Mod.III	Digitus	DGT-100
TRS-80 Mod.III	Digitus	DGT-1000
TRS-80 Mod.III	Kemitron	Naja 800
TRS-80 Mod.III	Prologica	CP-300
TRS-80 Mod.III	Prologica	CP-500
TRS-80 Mod.III	Sysdata	Sysdata III
TRS-80 Mod.III	Sysdata	Sysdata Jr.
TRS-80 Mod.IV	Multix	MX-Compacto
TRS-80 Mod.IV	Sysdata	Sysdata IV
TRS-Color	Codimex	CS-6508
TRS-Color	Dynacom	MX-1600
TRS-Color	LZ	Color 64
TRS-Color	Microdigital	TKS-800
TRS-Color	Prologica	CP-400

CLASSIFICAÇÃO DOS MICROCOMPUTADORES POR NOME

FABRICANTE	MODELO	PAÍS	LINHA
Appletronica	Thor 2010	Brasil	Apple II +
Apply	Apply 300	Brasil	Sinclair ZX-81
CCE	MC-4000 Exato	Brasil	Apple II +
CPA	Absolutus	Brasil	Apple II +
CPA	Polaris	Brasil	Apple II +
Codimex	CS-6508	Brasil	TRS-Color
Digitus	DGT-100	Brasil	TRS-80 Mod.III
Digitus	DGT-1000	Brasil	TRS-80 Mod.III
Digitus	DGT-AP	Brasil	Apple II +
Dismac	D-8000	Brasil	TRS-80 Mod. I
Dismac	D-8001/2	Brasil	TRS-80 Mod. I
Dismac	D-8100	Brasil	Apple II +
Dynacom	MX-1600	Brasil	TRS-Color
ENIAC	ENIAC II	Brasil	Apple II +
Engebras	AS-1000	Brasil	Sinclair ZX-81
Filcres	NEZ-8000	Brasil	Sinclair ZX-81
Franklin	Franklin	USA	Apple II +
Gradiente	Expert GPC1	Brasil	MSX
Houston	Houston AP	Brasil	Apple II +
Kemitron	Naja 800	Brasil	TRS-80 Mod.III
LNW	LNW-80	USA	TRS-80 Mod. I
LZ	Color 64	Brasil	TRS-Color
Magnex	DM II	Brasil	Apple II +
Maxitronica	MX-2001	Brasil	Apple II +
Maxitronica	MX-48	Brasil	Apple II +
Maxitronica	MX-64	Brasil	Apple II +
Maxitronica	Maxitronic I	Brasil	Apple II +
Microcraft	Craft II Plus	Brasil	Apple II +
Microcraft	Caft IIe	Brasil	Apple IIe
Microdigital	TK-3000 IIe	Brasil	Apple IIe
Microdigital	TK-82C	Brasil	Sinclair ZX-81
Microdigital	TK-83	Brasil	Sinclair ZX-81
Microdigital	TK-85	Brasil	Sinclair ZX-81
Microdigital	TK-90X	Brasil	Sinclair Spectrum
Microdigital	TKS-800	Brasil	TRS-Color
Milmar	Apple II Plus	Brasil	Apple II +
Milmar	Apple Master	Brasil	Apple II +
Milmar	Apple Senior	Brasil	Apple II +
Multix	MX-Compacto	Brasil	TRS-80 Mod.IV
Omega	MC-400	Brasil	Apple II +
Polymax	Maxxi	Brasil	Apple II +
Polymax	Poly Plus	Brasil	Apple II +
Prologica	CP-200	Brasil	Sinclair ZX-81
Prologica	CP-300	Brasil	TRS-80 Mod.III
Prologica	CP-400	Brasil	TRS-Color
Prologica	CP-500	Brasil	TRS-80 Mod.III
Ritas	Ringo R-470	Brasil	Sinclair ZX-81
Sharp	Hotbit HB-8000	Brasil	MSX
Spectrum	Microengenho I	Brasil	Apple II +
Spectrum	Microengenho II	Brasil	Apple IIe
Spectrum	Spectrum ed	Brasil	Apple II +
Suporte	Venus II	Brasil	Apple II +
Sycomig	SIC I	Brasil	Apple II +
Sysdata	Sysdata III	Brasil	TRS-80 Mod.III
Sysdata	Sysdata IV	Brasil	TRS-80 Mod.IV
Sysdata	Sysdata Jr.	Brasil	TRS-80 Mod.III
Timex	Timex 1000	USA	Sinclair ZX-81
Timex	Timex 1500	USA	Sinclair ZX-81
Timex	Timex 2000	USA	Sinclair Spectrum
Unitron	AP II	Brasil	Apple II +
Victor do Brasil	Elppa II Plus	Brasil	Apple II +
Victor do Brasil	Elppa Jr.	Brasil	Apple II +
Video Genie	Video Genie I	USA	TRS-80 Mod. I

UM LOGOTIPO PARA CADA MODELO DE COMPUTADOR

INPUT foi especialmente projetado para microcomputadores compatíveis com as sete principais linhas existentes no mercado.
Os blocos de textos e listagens de programas aplicados apenas a determinadas linhas de micros podem ser identificados por meio dos seguintes símbolos:

Sinclair ZX-81

TRS-80

TK-2000

MSX

Spectrum

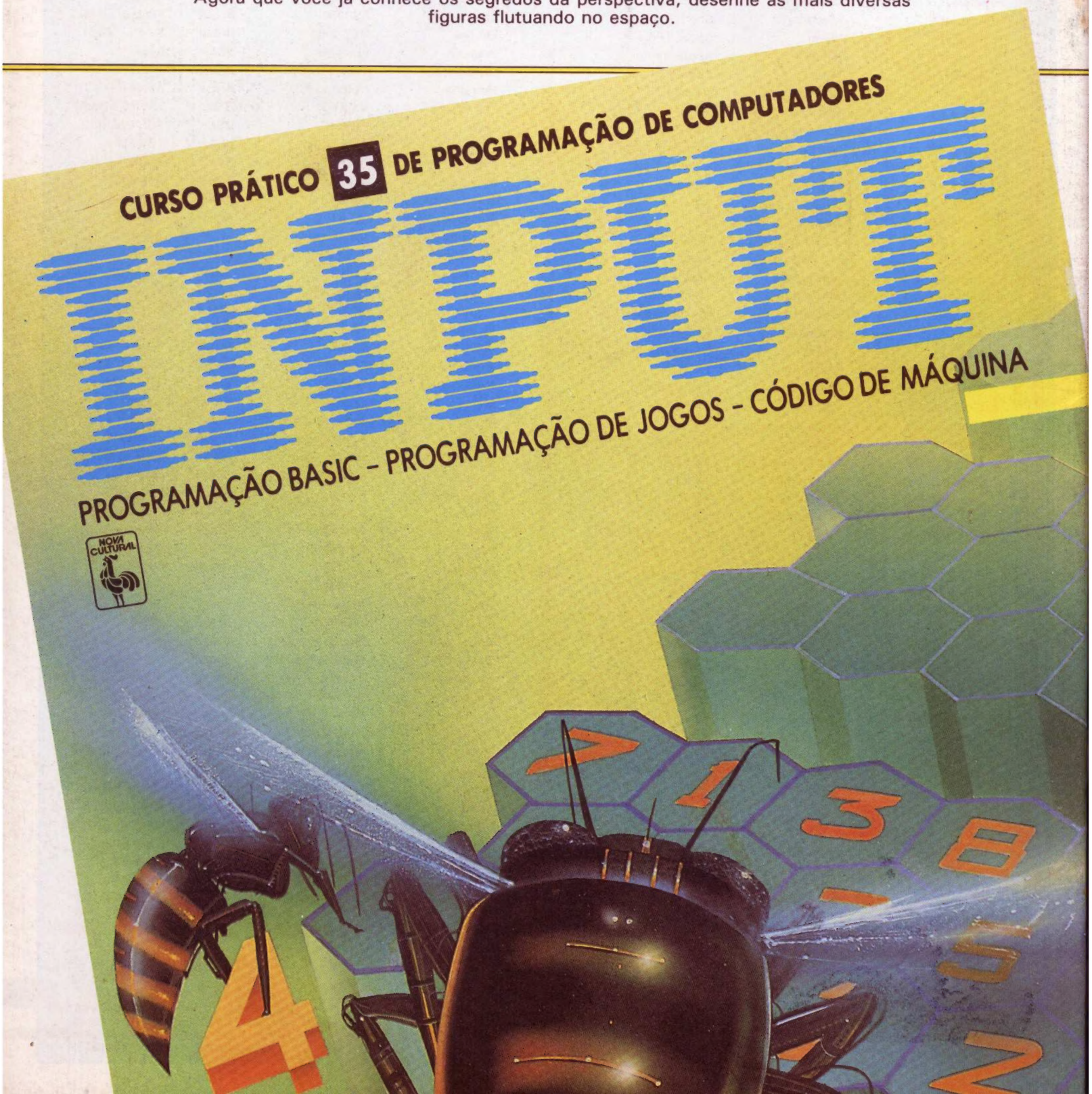
TRS-Color

Apple II

Quando o emblema for seguido de uma faixa, então tanto o texto como os programas que se seguem passam a ser específicos para a linha indicada.

NO PRÓXIMO NÚMERO

PROGRAMAÇÃO DE JOGOS
Para chegar ao ouro, você precisará fazer muitas pesquisas e escavações. Dinamize seu trabalho, recorrendo aos gráficos.

PROGRAMAÇÃO BASIC
Aprofunde seus conhecimentos sobre arquivos de dados e veja como transferir informações de um programa para outro.

PROGRAMAÇÃO BASIC
Agora que você já conhece os segredos da perspectiva, desenhe as mais diversas figuras flutuando no espaço.